DataSpace ACADEMY

Welcome to DataSpace Academy

DataSpace ACADEMY

Bipul Nath

Data Science Analyst at DataSpace Security Pvt. Ltd.

Machine Learning Engineer with more than 4 years experience in Data Science domain, proficient in predictive modeling, data processing, data anlysis and data visualization as well as python also I have knowledge in NLP, deep learning and computer vision. Developed various end to end machine learning and deep learning project like DETECTING THE COVID-19 CASES FROM CHEST X-RAY IMAGES, TOMATO LEAF DISEASE DETECTION, CHRONIC - KIDNEY - DISEASE PREDICTION, REAL TIME SPAM DETECTION.

DataSpace ACADEMY

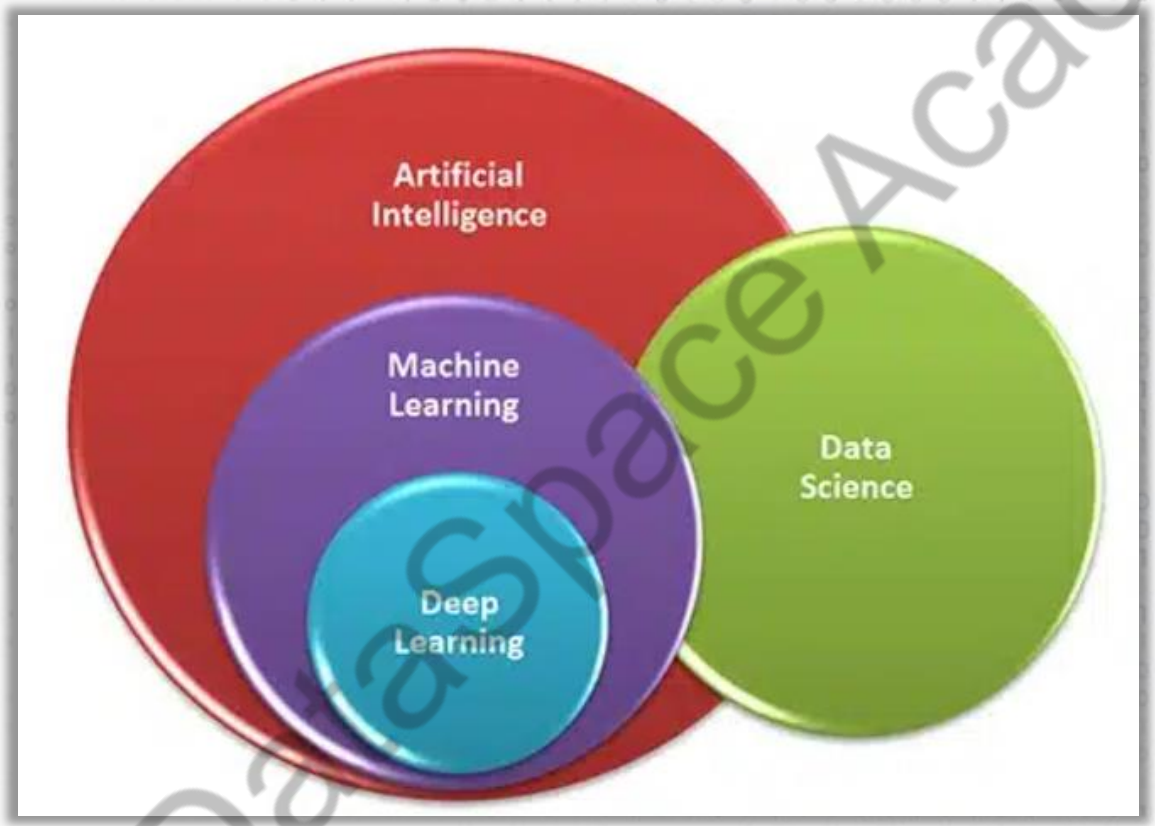
3

DATA SCIENCE

Welcome to Data Science Workshop

CONTENTS

ML VS DS VS AI



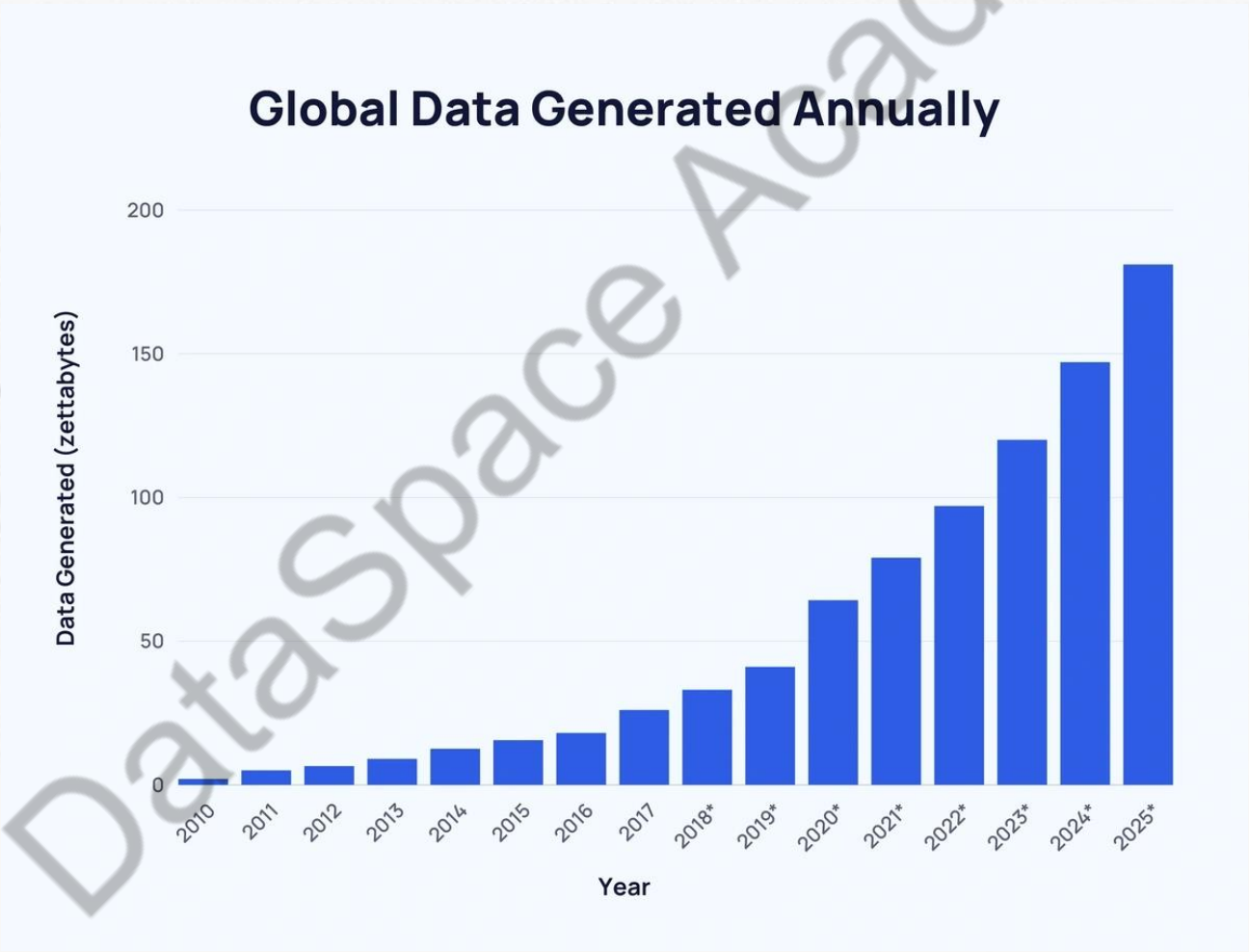
ML VS DL VS AI CONTD.

- **Artificial intelligence (AI)** is related to making machines intelligent and make them perform human tasks. For example self driving car.
- **Machine Learning** is an application of AI. ML is concept which allows machine to learn from example and experience.
- **Deep Learning** is a part of ML which is applied to larger and complex data-sets, it tries to mimic human brain.
- **Data science** is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data.
- **Big Data** is if more data getting generated at high speed with many variety and trustworthy then it's called Big Data.

WHAT IS DATA SCIENCE ?

- Data Science is the vast field.
- Data Science is the Process of collecting the data, storing the data, preprocessing the data which basically means to remove the unnecessary data and then using some tools and techniques we analyze the data and finally apply decision making based upon the data we collect. This whole process known as data science.
- Data Science is the science of data study using statistics, models, algorithms that is used by companies /industries to gain insights and to predict future.
- Data Science is a more reliable focused theme that predicts answers for a long term goals.

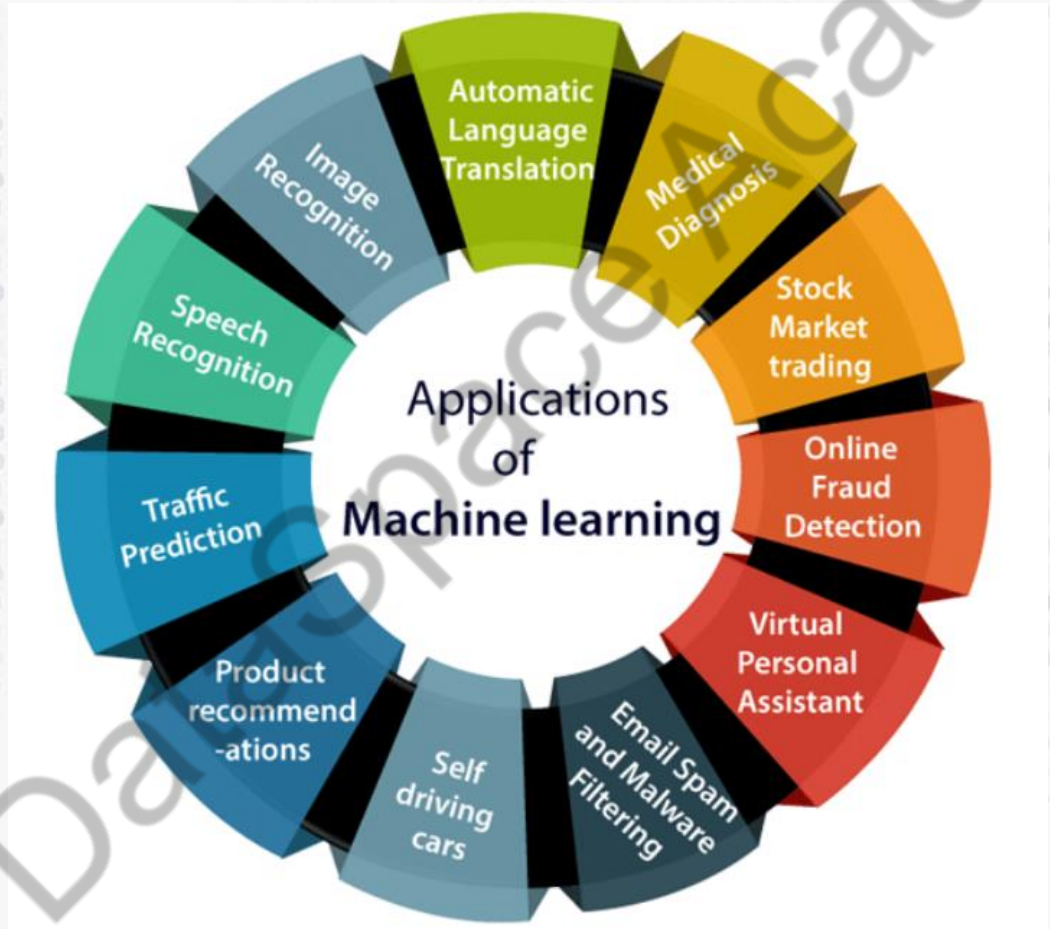
DATA GENERATED OVER THE YEAR



WHY DATA SCIENCE

- Data Science is the next and most important added value for business.
- Data science is exciting - data fuels the future
  - Data has been called "the oil of the digital economy" because of its immense potential.
  - Data analyst provides completely new generations of technological solutions
  - Various business sectors such as energy, government, healthcare, intelligence, manufacturing, retail, textiles, automobile, banking ,security, and logistics are using their data to improve their business decision.

APPLICATION OF DATA SCIENCE



APPLICATION CONTD.

- A spam ham detector is a program that can automatically classify a message as spam or ham. Spam is unsolicited or unwanted electronic messages, typically sent in bulk. Ham is any message that is not spam.

APPLICATION CONTD.

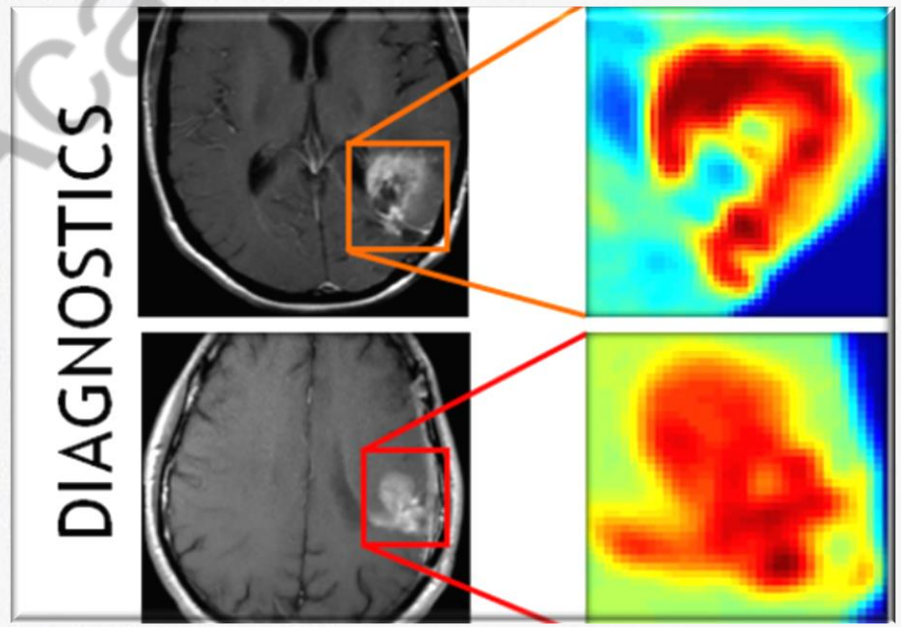
- **HR Analytics -**
- Companies use models to predict the productivity the new employee based on the past data training on the model .
- Companies can select 50 employees among 900 resumes with assistance of Data science

APPLICATION CONTD.

- Data science can be used to automate fraud detection. This can free up human resources so that they can focus on other tasks. Additionally, automation can help to ensure that fraud detection is performed consistently and efficiently.

APPLICATION CONTD.

- Machine learning algorithms can aid in diagnosing medical conditions by analyzing medical images (such as X-rays, MRIs, and CT scans), identifying patterns in genomic data, and predicting disease progression. Additionally, data-driven insights can guide personalized treatment plans based on patient characteristics and historical treatment outcomes.



APPLICATION CONTD.

- **NLP** (Natural Language Processing)
- Chatbot
- ChatGPT

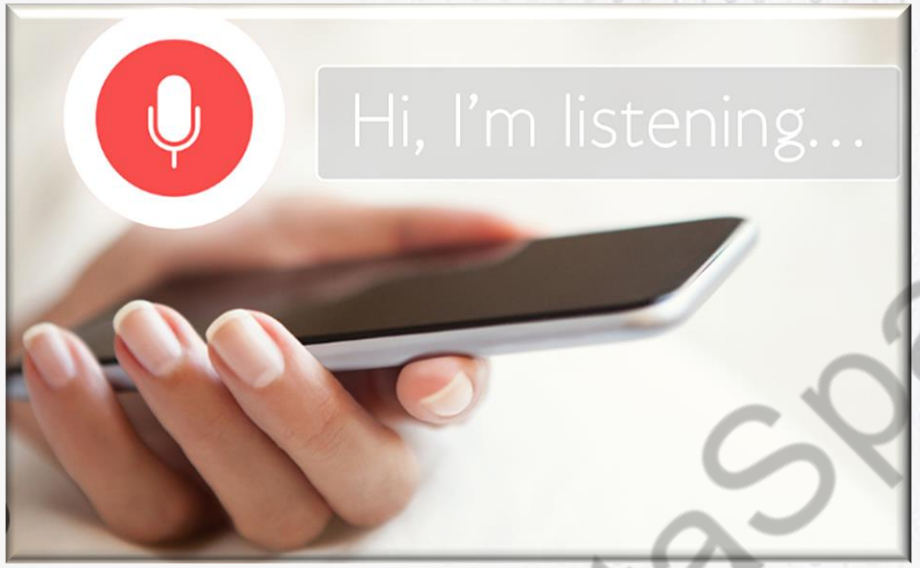
APPLICATION CONTD.

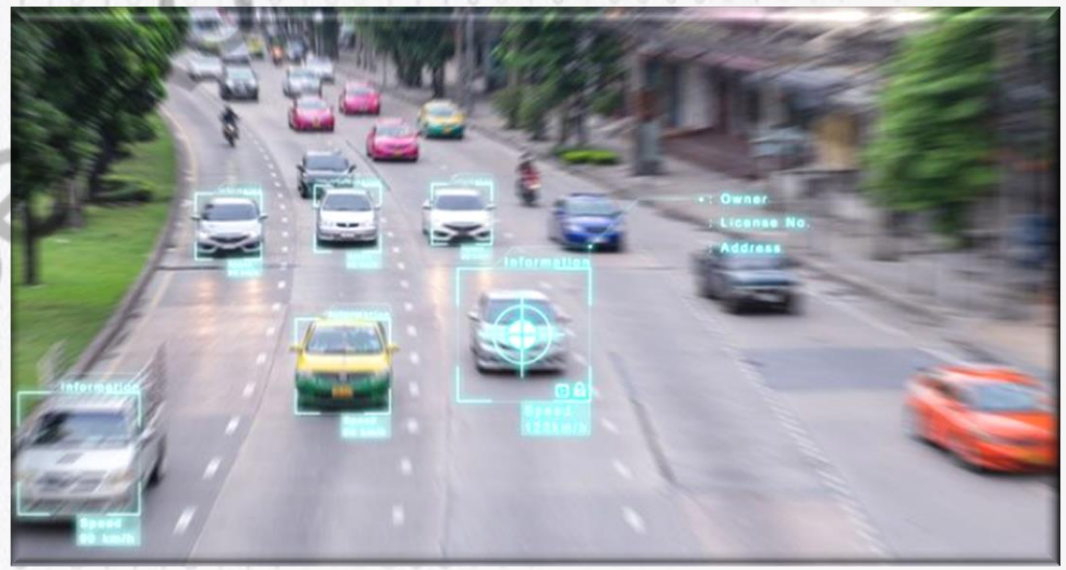
- Data science is being increasingly used in the stock market to help traders and investors make better decisions.

APPLICATION CONTD.

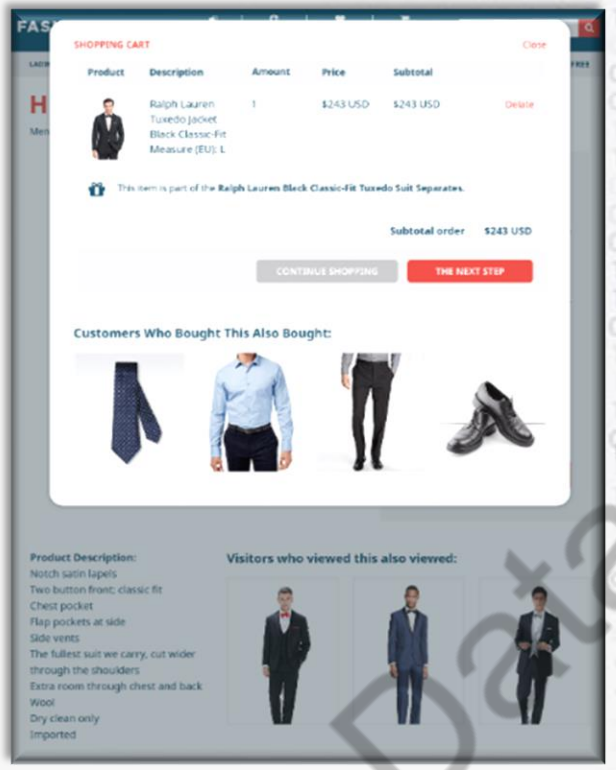
- Facebook uses a tool called DeepFace for face recognition. DeepFace is a deep learning algorithm that has been trained on a massive dataset of images. It is able to recognize faces with an accuracy of over 99%.

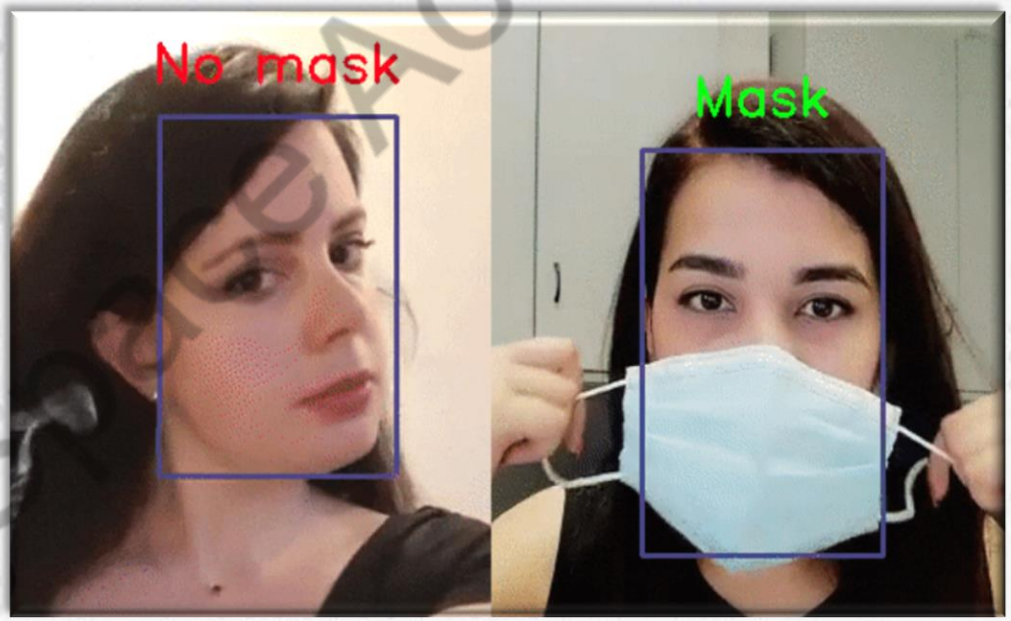
APPLICATION CONTD.





APPLICATION CONTD.

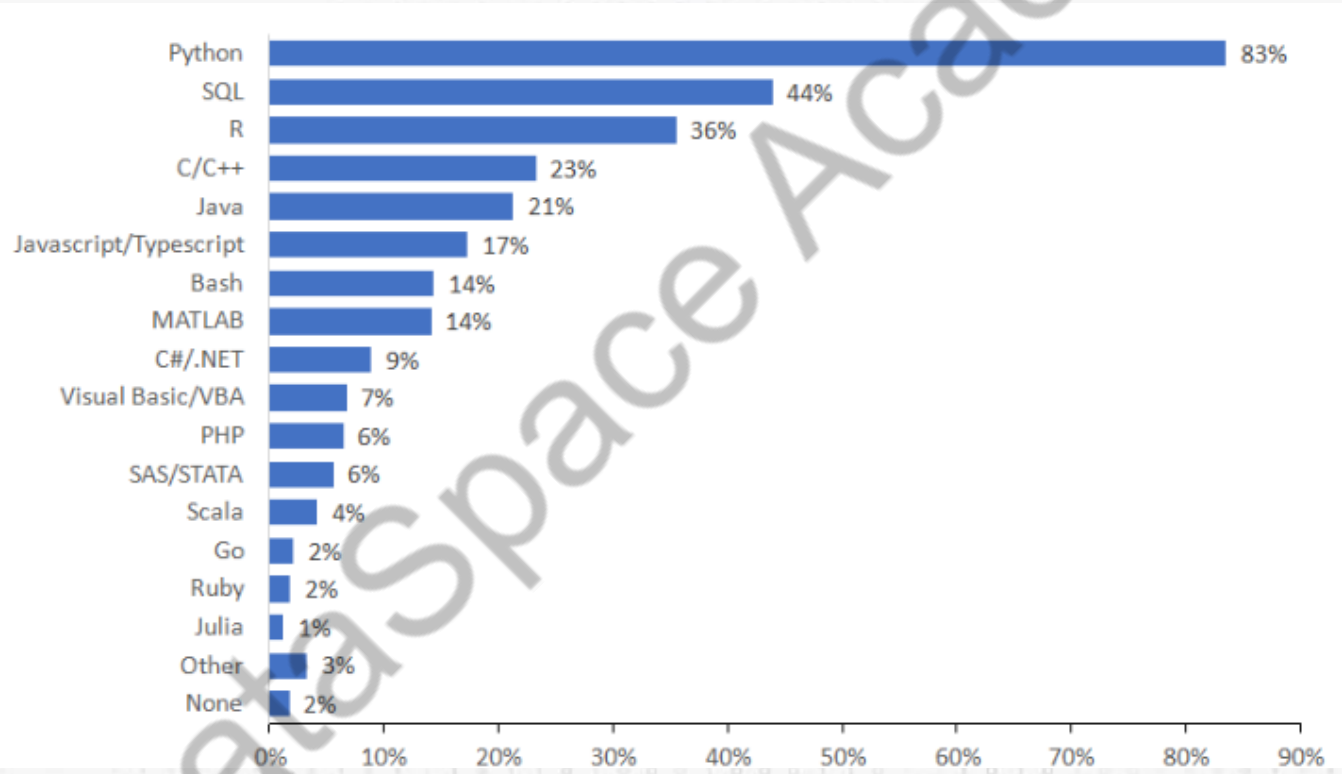




BENEFITS OF DATA SCIENCE IN INDUSTRIES

- Cost Reduction
- Faster and Better Decision making
- Improved Services and Products
- Next Generation Products

LANGUAGE COMPARISON



DataSpace ACADEMY

PYTHON PROGRAMMING

Welcome to Python Programming

22

INDEX

INTRODUCTION

- Easy to learn – Beginner's programming language
- Cross-platform Interpreted Language
- Clean and Elegant Syntax
- Free and Open source
- Object-Oriented Programming Language
- Large Standard Library
- Integrated – Easily integrated with C, C++, JAVA, etc..
- Magic Methods

PYTHON HISTORY

- Created by **Guido Van Rossum.**
- The idea of python started in early 1980
- Finally published in on February 20, 1991
- PYTHON 1.0 release in the year of 1994
- PYTHON 2.0 release in the year of 2000
- PYTHON 3.0 release in the year of 2008
- The inspiration for the name came from BBC's TV Show –

  'Monty Python's Flying Circus'

APPLICATION AREA

- **Web Development** – Django, Flask, Web2py, Bottle, Tornado

APPLICATION AREA

- **Graphical User Interfaces** – Kivy, PyQT, Tkinter, PyGUI, PySide, WxPython, pygame

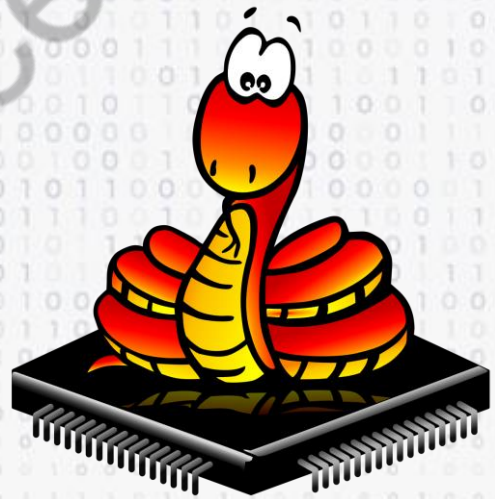
APPLICATION AREA

- **Multimedia Programming -** PyMedia, GStreamer, PyGame, Pyglet

pyglet

APPLICATION AREA

- **Database Programming –** Supporting connectors for all databases
- **Networking –** Asyncio, Diesel, Pulsar, Twisted, NAPALM
- **Automation –** ANN, IOT, MicroPython, Embedded Systems



APPLICATION AREA

- **Web Scraping –** Beautiful Soup, Scrapy, Requests, Selenium
- **System Administration –** Fabric, Salt, Psutil, Ansible, Chef, Puppet, Blueprint, Buildout, Shinken
- **Scientific Computing –** Astrophy, Biopython, Bokeh, Cubes, Dask, Matplotlib, NetworkX, NumPy, Pandas, PyTorch, SciPy

APPLICATION AREA

- **Text Processing –** NLTK, Polyglot, TextBlob, CoreNLP, Pattern, Vocabulary, QuePy
- **Image Processing –** OpenCV, Pillow, Scikit-image
- **Machine Learning –** Scikit-learn, TensorFlow, Keras, Theano, PyTorch
- **Data Analytics –** NumPy, Pandas, Matplotlib, Seaborn, Plot.ly, SciPy
- **Data Science –** ML, DA, TensorFlow, Keras

LETS DO SOME CODING

- 4+5
- 8/3
- 4//2
- 19* “o”
- a = 34
- b = “good”
- print(b)
- print(a,b)

GETTING OUTPUT IN CONSOLE

- The print() function
  - Syntax
    - print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
  - *objects – Objects to print (can be single or multiple).
  - sep – The separator string to separate two values in case of printing multiple objects.
  - end – String to be printed after all objects are printed.
  - file – The location where we want to print our output.
  - flush – Normally output to a file or the console is buffered, with text output at least until you print a newline. The flush makes sure that any output that is buffered goes to the destination.

PYTHON COMMENTS

- Comments can be used to explain Python code.
- Comments can be used to make the code more readable.
- Comments can be used to prevent execution when testing code.
- Comments starts with a #, and Python will ignore them

```
#This is a comment
print("Hello, World!")
print("Hello, World!") #This is a comment
```

- We can use multi line comments also using triple quoted string (although it means something else, we will discuss it later on)

```
"""This is a comment
    Multiline comment"""
```

PYTHON VARIABLES

- Variable is a name used to refer memory locations because we can't remember memory addresses and used to hold values.
- In Python, we don't need to specify type of the variable because Python is smart enough to get variable type.
- Variables can be named using combination of letters, digits and underscore (_) . No other special symbol can be used.
- Variables can begin with letter or underscore but not with digit.
- Python is case sensitive language.
- Variables can also be called as identifiers.
- Valid identifiers:- a123, _n, n_9
- Invalid identifiers:- 1a, n%4, n 9
- The equal (=) operator is used to assign value to a variable.
- Python doesn't bound us to declare variables before use. It allows us to create variable at required time.
- Example:

```
a = 10
name = "Python Developer"
```

PYTHON VARIABLES CONTD.

- Python support Multiple Assignments

```
x=y=z=50    #(Here, x,y,z have same value i.e., 50)
x,y=20,42     #(Here, x =20 and y=42)
```

- Examples to demonstrate the variables: -

```
a=10
b,c=20,30
x=y=z=50
print(a)
print(b)
print(c)
print(x)
print(y)
print(z)
```

PYTHON KEYWORDS

```
>>> import keyword

>>> keyword.kwlist

['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class',
'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for',
'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal',
'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with',
'yield']
```

PYTHON OPERATORS

- Arithmetic `+,-,*,/,//,**,%`
- Relational `>, =, <=`
- Assignment `=, +=, -=, *=, /=, //=, **=, %=`
- Logical `and, or, not`
- Bitwise `&, |, ^, <>`
- Identity `is, is not`
- Membership `in, not in`

USE OF BRACKETS

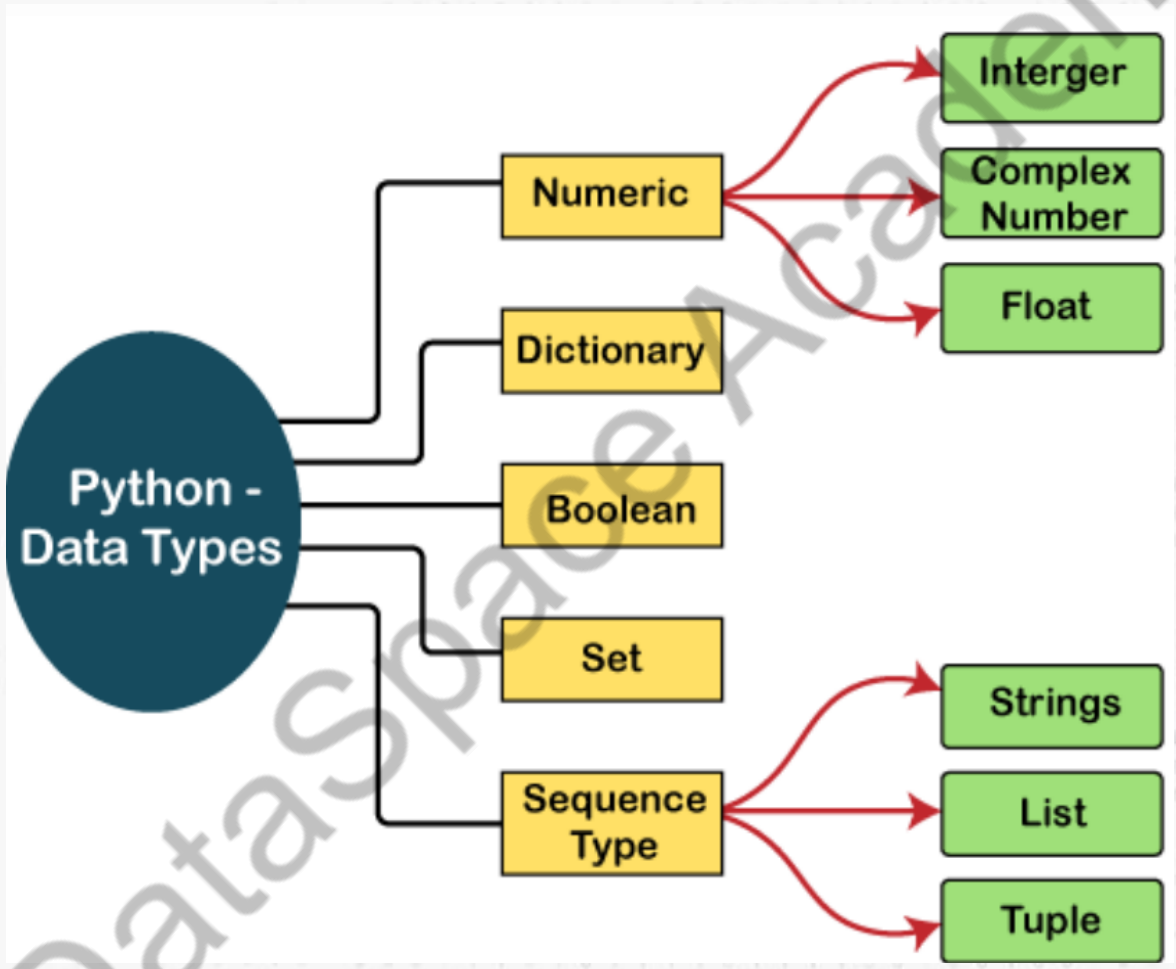
- () Function Arguments, Expressions, Tuples
- {} Dictionary, Sets
- [] Lists, Indexing Slicing
- <> not used as brackets

DataSpace ACADEMY

40

DATA TYPES

Python Built-in Data Types



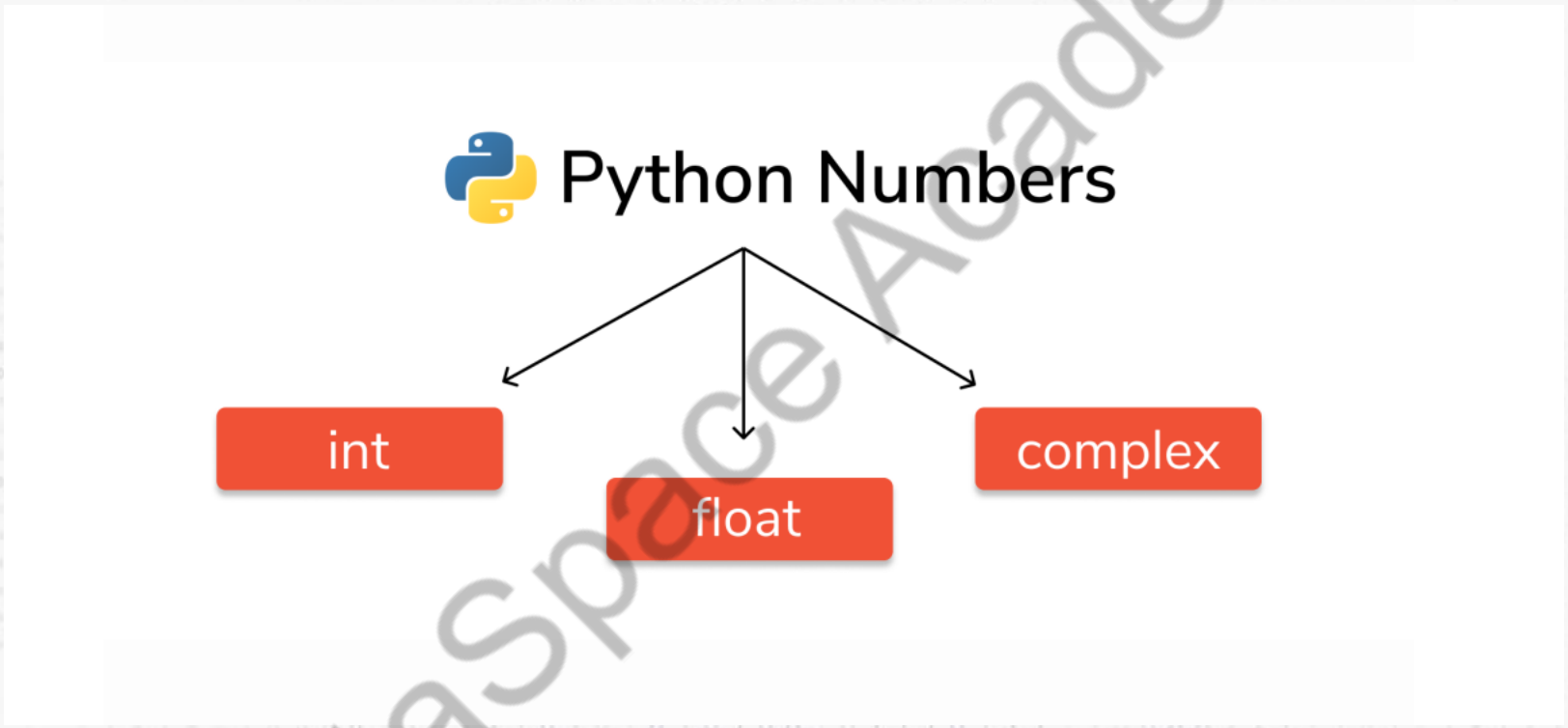
Name	Type	Description
Integers	int	Whole numbers, such as: **3 300 200**
Floating point	float	Numbers with a decimal point: **2.3 4.6 100.0**
Strings	str	Ordered sequence of characters: **"hello" 'Sammy' "2000" "楽しい"**
Lists	list	Ordered sequence of objects: **[10,"hello",200.3]**
Dictionaries	dict	Unordered Key:Value pairs: **{"mykey" : "value" , "name" : "Frankie"}**
Tuples	tup	Ordered immutable sequence of objects: **(10,"hello",200.3)**
Sets	set	Unordered collection of unique objects: **{"a","b"}**
Booleans	bool	Logical value indicating **True** or **False**

DataSpace ACADEMY

43

NUMERIC CATEGORIES

Data Classes Built-in Functions



LETS DO CODING...

DataSpace ACADEMY

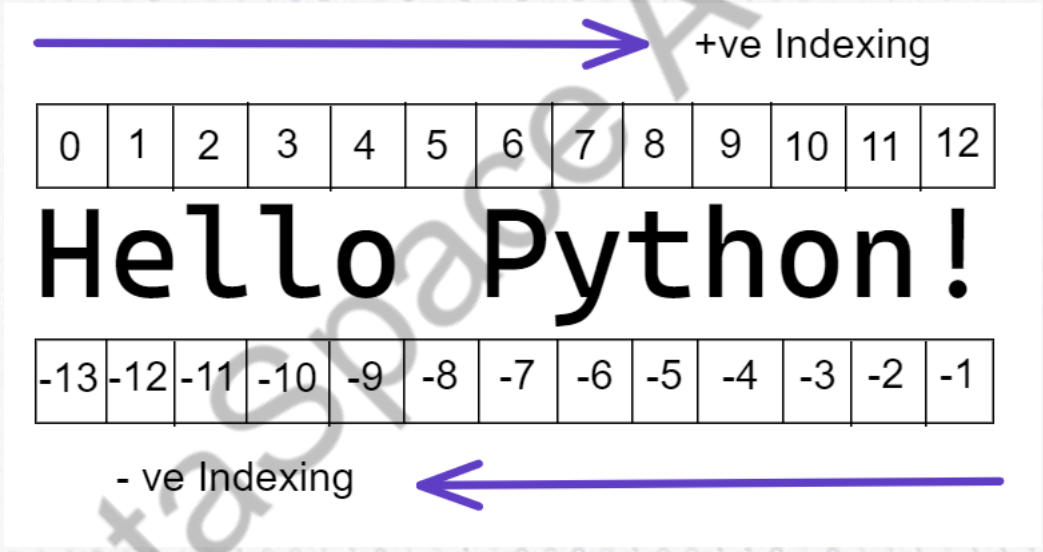
46

STRINGS

Data Classes Built-in Functions

ACCESSING STRING USING INDEXING

- Index of strings starts with 0 and ends with (n-1). Here, n stands for the total number of characters in the string i.e., length of string.



BUILT-IN FUNCTIONS AVAILABLE ON STRINGS

- capitalize()
- title()
- upper()
- lower()
- center()
- split()
- count()
- len()
- replace()

LETS DO CODING...

DataSpace ACADEMY

LIST

Data Classes Built-in Functions

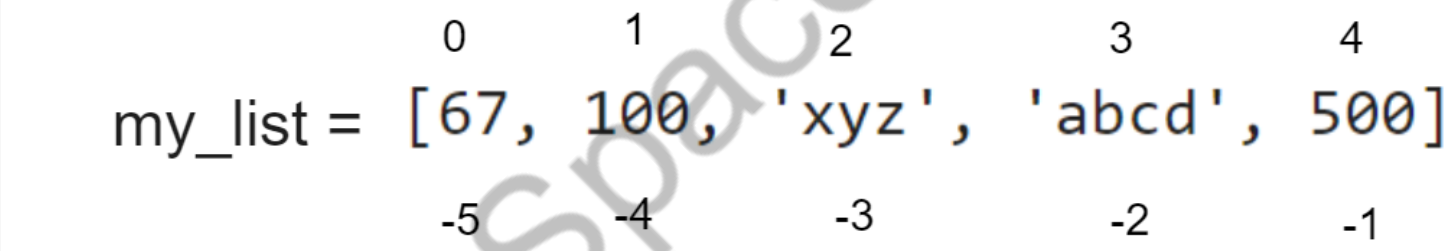
LIST

- List Concatenation
- List Reptation

ACCESSING LIST USING INDEXING AND SLICING

DataSpace ACADEMY

- Index of lists starts with 0 and ends with (n-1). Here, n stands for the total number of elements in the list i.e., length of list.



LIST FUNCTIONS

- len()
- append()
- extend()
- remove()
- count()
- reverse()

LETS DO CODING...

DataSpace ACADEMY

55

DICTIONARY

Data Classes Built-in Functions

ACCESSING DICTIONARY

- To access dictionary value use key enclosed by square braces.
- Example
- d = {“name”:”Akash”,“age”:22}
- print(d[“name”])

DICTIONARY FUNCTIONS

- len()
- pop()
- keys()
- values()
- items()
- copy()
- clear()

LETS DO CODING...

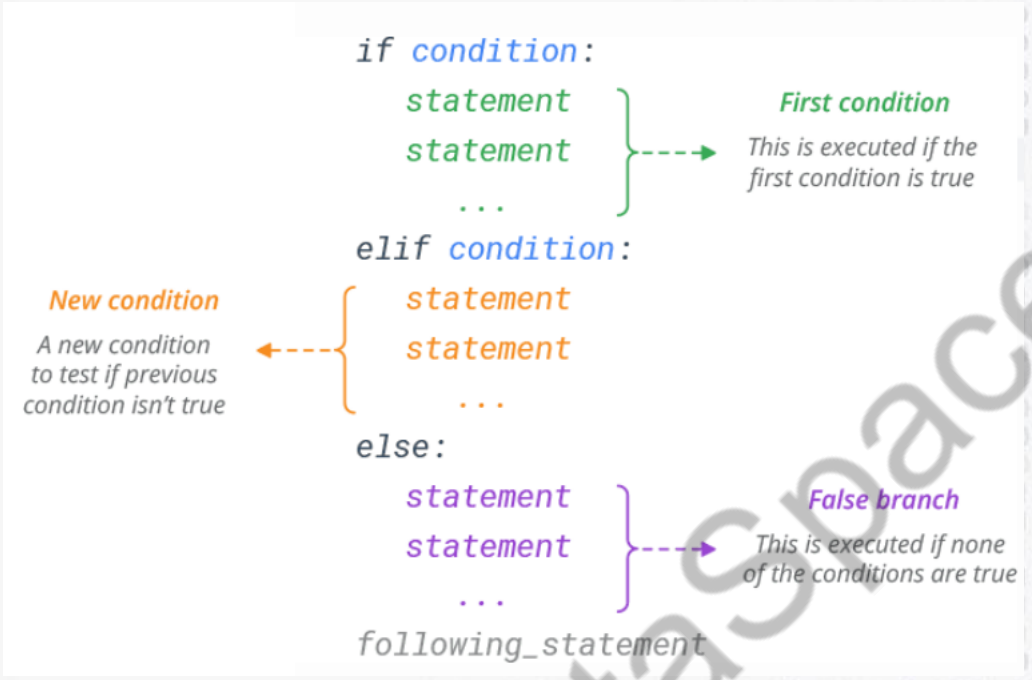
DataSpace ACADEMY

59

CONTROL FLOW STATEMENTS

Python Programming

IF-ELIF-ELSE




```
x, y = 5, 5
if x > y:
    print('x is greater')
elif x < y:
    print('y is greater')
else:
    print('x and y are equal')

# Prints x and y are equal
```

LETS DO CODING...

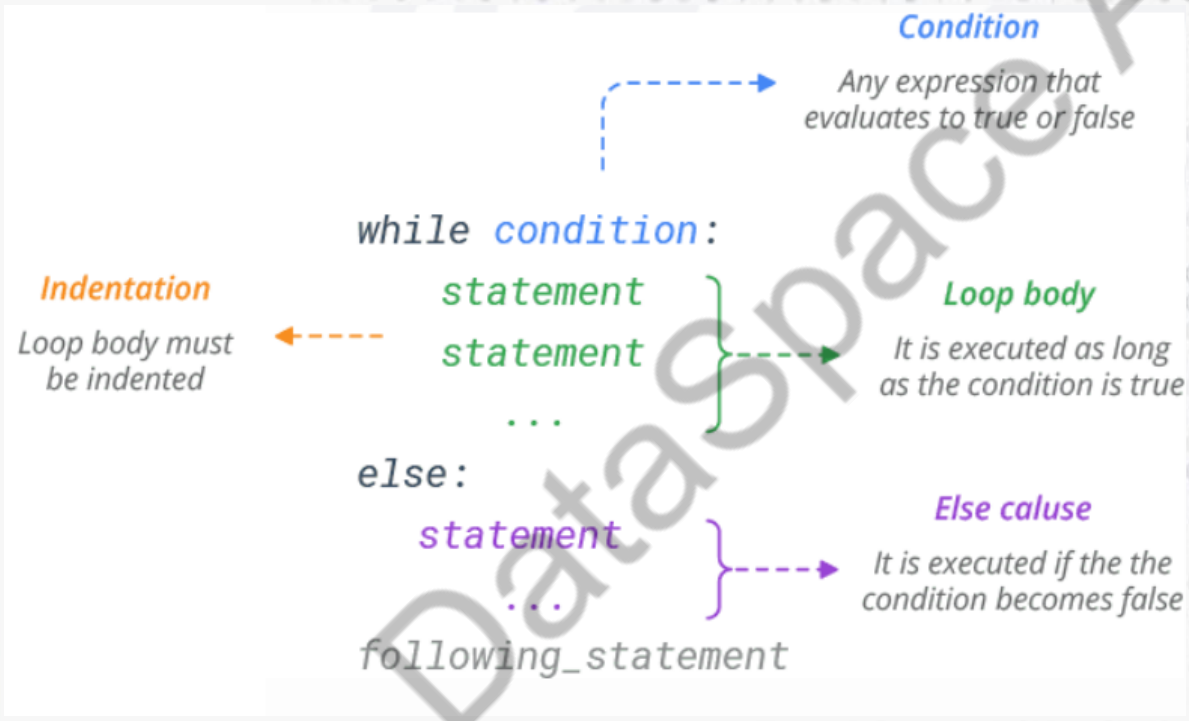
DataSpace ACADEMY

WHILE

Loop Statements

WHILE

- A while loop is used when you want to perform a task indefinitely, until a particular condition is met. It's a condition-controlled loop.




```
# Iterate until string is empty
x = 'blue'
while x:
    print(x)
    x = x[1:]
# Prints blue
# Prints lue
# Prints ue
# Prints e
```

WHILE LOOP EXAMPLES

```
# Iterate until x becomes 0
x = 6
while x:
    print(x)
    x -= 1
# Prints 6 5 4 3 2 1
```

```
# Iterate until list is empty
L = ['red', 'green', 'blue']
while L:
    print(L.pop())
# Prints blue green red
```

```
# Iterate until string is empty
x = 'blue'
while x:
    print(x)
    x = x[1:]
# Prints blue
# Prints lue
# Prints ue
# Prints e
```

```
# Exit condition is false at the start
x = 0
while x:
    print(x)
    x -= 1
```

INFINITE LOOP

```
# Infinte loop with while statement
while True:
    print('Press Ctrl+C to stop me!')
```

```
# Loop runs until the user enters 'stop'
while True:
    name = input('Enter name:')
    if name == 'stop': break
    print('Hello', name)

# Output:
# Enter name:Bob
# Hello Bob
# Enter name:Sam
# Hello Sam
# Enter name:stop
```

DataSpace ACADEMY

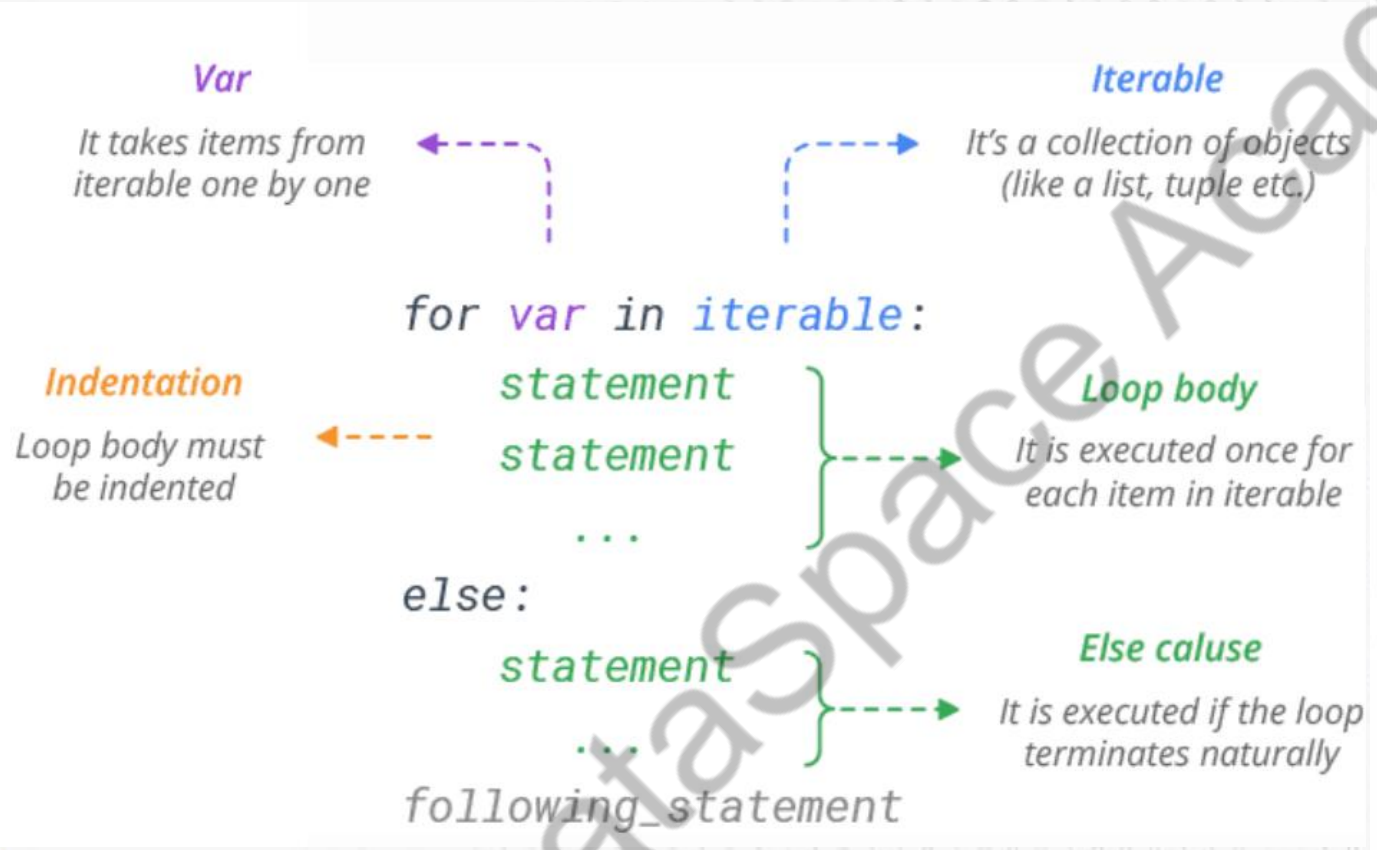
FOR

Loop Statements

FOR LOOP

- The for statement in Python is a bit different from what you usually use in other programming languages.
- Rather than iterating over a numeric progression, Python's for statement iterates over the items of any iterable (list, tuple, dictionary, set, or string).
- The items are iterated in the order that they appear in the iterable.

FOR LOOP CONTD.




```
# Iterate through a list
colors = ['red', 'green', 'blue', 'yellow']
for x in colors:
    print(x)
# Prints red green blue yellow
```

```
# Iterate through a string
S = 'python'
for x in S:
    print(x)
# Prints p y t h o n
```

```
# Flatten a nested list
list = [[1, 2, 3],[4, 5, 6],[7, 8, 9]]
for sublist in list:
    for number in sublist:
        print(number)
# Prints 1 2 3 4 5 6 7 8 9
```

BREAK & CONTINUE IN FOR LOOP

```
# Break the loop at 'blue'
colors = ['red', 'green', 'blue', 'yellow']
for x in colors:
    if x == 'blue':
        break
    print(x)
# Prints red green
```

```
# Skip 'blue'
colors = ['red', 'green', 'blue', 'yellow']
for x in colors:
    if x == 'blue':
        continue
    print(x)
# Prints red green yellow
```

USING RANGE FUNCTIONS

- The range(start,stop,step) function generates a sequence of numbers from 0 up to (but not including) specified number.

```
# Print 'Hello!' three times
for x in range(3):
    print('Hello!')
# Prints Hello!
# Prints Hello!
# Prints Hello!
```

```
# Generate a sequence of numbers from 2 to 6
for x in range(2, 7):
    print(x)
# Prints 2 3 4 5 6
```

```
for x in range(-5,0):
    print(x)
# Prints -5 -4 -3 -2 -1
```

```
# Increment the range with 2
for x in range(2, 7, 2):
    print(x)
# Prints 2 4 6
```

LETS DO CODING...

DataSpace ACADEMY

72

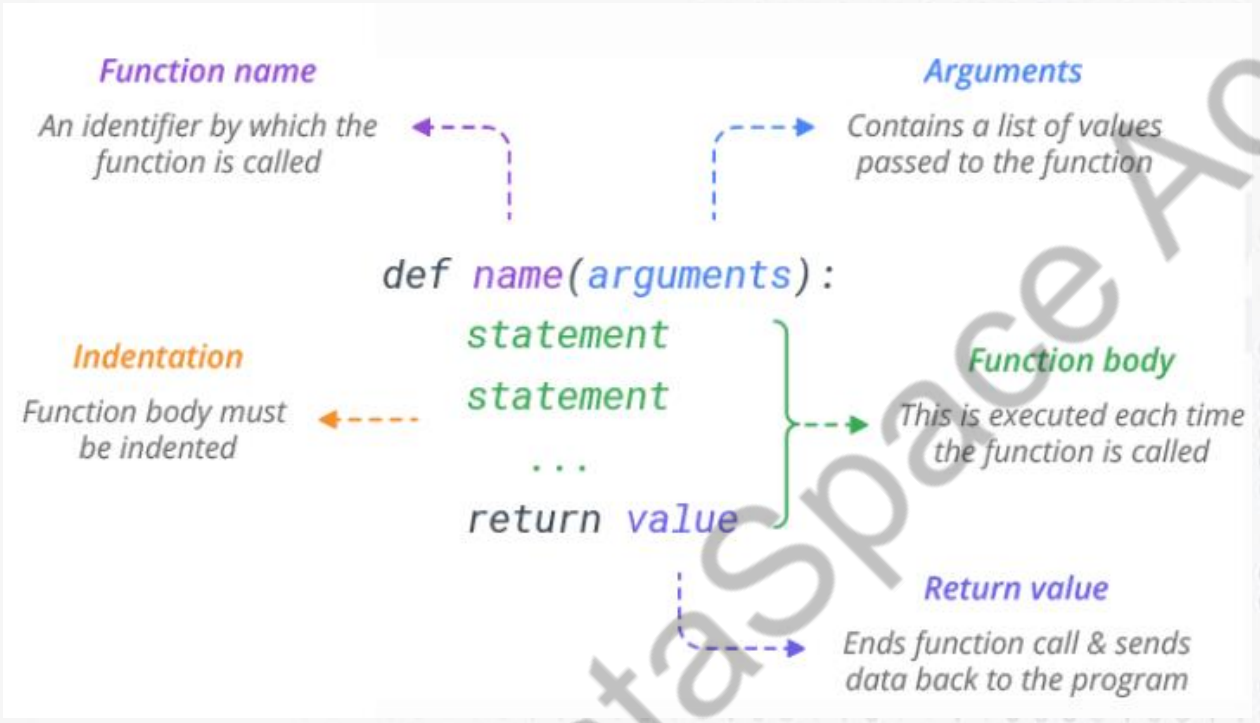
USER DEFINED FUNCTIONS

Python Functions

PYTHON USER DEFINED FUNCTIONS

- A block of code which only runs when it is called.
- Enables code reusability.
- We can pass data, known as parameters.
- A function can return data as a result.
- We use def keyword to create a function.
- A block is created, So pay attention to Indentations.
- Build to perform a specific task.
- Types of Functions
  - Parameterized and Non Parameterized functions
  - Anonymous Functions/Lambda Expressions
  - Generator Functions
  - Decorator Functions

UDF



- Creating

```
def hello():
    print('Hello, World!')
```

- Calling

```
hello()
# Prints Hello, World!
```

RETURN VALUE FROM FUNCTIONS

- To return a value from a function, simply use a return statement.
- Once a return statement is executed, nothing else in the function body is executed.

```
# Return sum of two values
def sum(a, b):
    return a + b

x = sum(3, 4)
print(x)
# Prints 7
```

- Python has the ability to return multiple values, something missing from many other languages.
- You can do this by separating return values with a comma

```
# Return addition and subtraction in a tuple
def func(a, b):
    return a+b, a-b

result = func(3, 2)

print(result)
# Prints (5, 1)
```

THE DOCSTRING

- You can attach documentation to a function definition by including a string literal just after the function header.
- Docstrings are usually triple quoted to allow for multi-line descriptions.

```
def hello():
    """This function prints
       message on the screen"""
    print('Hello, World!')
```

```
# Print docstring in rich format
help(hello)

# Help on function hello in module __main__:
# hello()
#     This function prints
#     message on the screen
```

```
# Print docstring in a raw format
print(hello.__doc__)

# Prints This function prints message on the screen
```

FUNCTIONS OPERATIONS

- Nested Functions

```
def outer(a, b):
    def inner(c, d):
        return c + d
    return inner(a, b)

result = outer(2, 4)

print(result)
# Prints 6
```

- Recursion

```
def countdown(num):
    if num <= 0:
        print('Stop')
    else:
        print(num)
        countdown(num-1)

countdown(5)
# Prints 5
# Prints 4
# Prints 3
# Prints 2
# Prints 1
# Prints Stop
```

- Assigning function to variable

```
def hello():
    print('Hello, World!')

hi = hello
hi()
# Prints Hello, World!
```

LETS DO CODING...

LAMBDA EXPRESSION

- A lambda is simply a way to define a function in Python. They are sometimes known as lambda expressions or lambda operators.
- The lambda functions are anonymous. Meaning, they are functions that do not need to be named. They are used to create small one-line functions in cases where a normal function would be an overkill
- Have any number of arguments but have only one expression.
- Always return an expression

lambda parameters: expression

LAMBDA FUNCTION CONTD.

```
def doubler(x):
    return x*2

print(doubler(2))
# Prints 4

print(doubler(5))
# Prints 10
```

```
doubler = lambda x: x*2

print(doubler(2))
# Prints 4

print(doubler(5))
# Prints 10
```

```
evenOdd = (lambda x:
           'odd' if x%2 else 'even')

print(evenOdd(2))
# Prints even

print(evenOdd(3))
# Prints odd
```

```
# A lambda function that adds three values
add = lambda x, y, z: x+y+z
print(add(2, 5, 10))
# Prints 17
```

DataSpace ACADEMY

81

MODULE AND PACKAGES

Python Modules

INSTALLING PYTHON MODULES

- python –m pip install
- Or
- pip install <module>
- For multiple python versions, specify python version
- To install PIP (Python Package Installer)
  - Download get-pip.py
  - Run
    - python get-pip.py

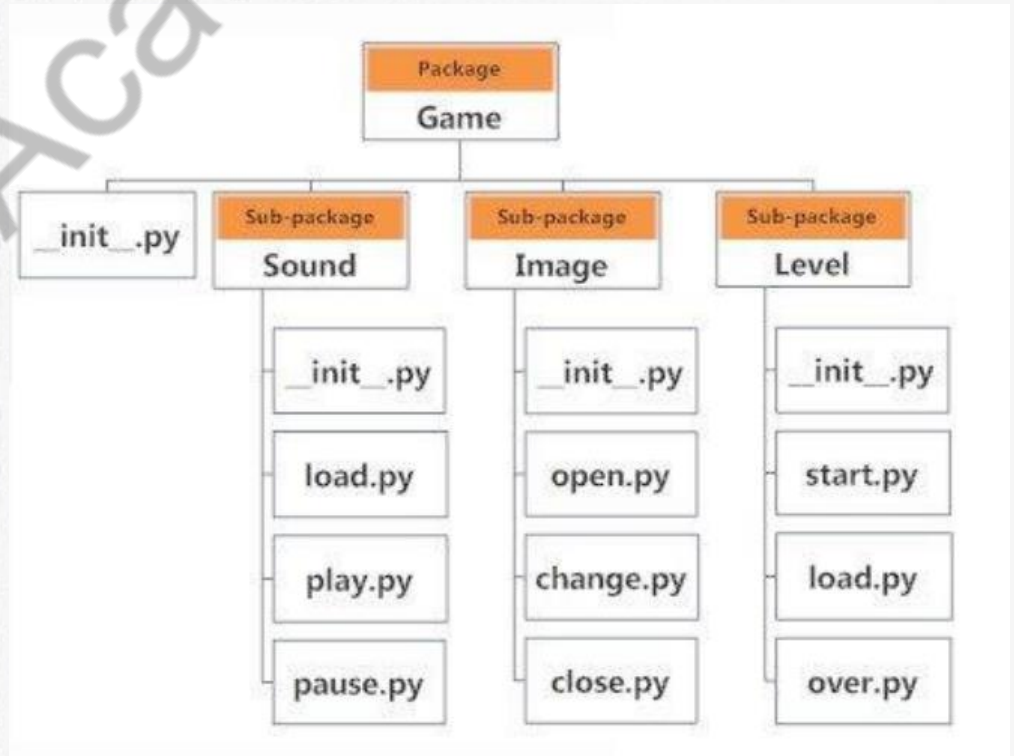
COMMON MODULES

- os
- sys
- math
- json
- csv
- numpy
- pandas
- sklearn
- tkinter

```
>>> import fibo, sys
>>> dir(fibo)
['__name__', 'fib', 'fib2']
>>> dir(sys)
['__breakpointhook__', '__displayhook__', '__doc__', '__excepthook__',
 '__interactivehook__', '__loader__', '__name__', '__package__', '__spec__',
 '__stderr__', '__stdin__', '__stdout__', '__unraisablehook__',
 '_clear_type_cache', '_current_frames', '_debugmallocstats', '_framework',
 '_getframe', '_git', '_home', '_xoptions', 'abiflags', 'addaudithook',
 'api_version', 'argv', 'audit', 'base_exec_prefix', 'base_prefix',
 'breakpointhook', 'builtin_module_names', 'byteorder', 'call_tracing',
 'callstats', 'copyright', 'displayhook', 'dont_write_bytecode', 'exc_info',
 'excepthook', 'exec_prefix', 'executable', 'exit', 'flags', 'float_info',
 'float_repr_style', 'get_asyncgen_hooks', 'get_coroutine_origin_tracking_depth',
 'getallocatedblocks', 'getdefaultencoding', 'getdlopenflags',
 'getfilesystemencodeerrors', 'getfilesystemencoding', 'getprofile',
 'getrecursionlimit', 'getrefcount', 'getsizeof', 'getswitchinterval',
 'gettrace', 'hash_info', 'hexversion', 'implementation', 'int_info',
 'intern', 'is_finalizing', 'last_traceback', 'last_type', 'last_value',
 'maxsize', 'maxunicode', 'meta_path', 'modules', 'path', 'path_hooks',
 'path_importer_cache', 'platform', 'prefix', 'ps1', 'ps2', 'pycache_prefix',
 'set_asyncgen_hooks', 'set_coroutine_origin_tracking_depth', 'setdlopenflags',
 'setprofile', 'setrecursionlimit', 'setswitchinterval', 'settrace', 'stderr',
 'stdin', 'stdout', 'thread_info', 'unraisablehook', 'version', 'version_info',
 'warnoptions']
```

PYTHON PACKAGES

- Packages are a way of structuring Python's module namespace by using "dotted module names".
- Similar files are kept in the same directory, for example, we may keep all the songs in the "music" directory. Analogous to this, Python has packages for directories and modules for files.
- As our application program grows larger in size with a lot of modules, we place similar modules in one package and different modules in different packages. This makes a project (program) easy to manage and conceptually clear.
- Similarly, as a directory can contain subdirectories and files, a Python package can have sub -packages and modules.
- A directory must contain a file named __init__.py in order for Python to consider it as a package. This file can be left empty but we generally place the initialization code for that package in this file.



DataSpace ACADEMY

85

DAY 2

DataSpace ACADEMY

86

PANDAS

Data Analysis Tool

WHAT IS PANDAS ?

- Pandas is a Python library used for working with datasets.
- It has functions for analyzing, cleaning, exploring, and manipulating data.
- Pandas allows importing data from various file formats such as comma-separated values, tab-separated values, JSON, SQL database tables or queries, and Microsoft Excel.
- Even we can do some basic visualization with it.

TYPES OF DATA STRUCTURE IN PANDAS

DataSpace ACADEMY

Data Structure	Dimension	Description
Series	Single column	It is a 1D array holding data of any type. Example.
DataFrame	Multiple column	It is a 2D data structure, like a 2D array, or a table with rows and columns.
Panel	Multiple Table	It is a 3D container of data.

PANDAS SERIES

- Pandas Series is a one-dimensional labeled array capable of holding any data type, like int, float, string etc.
- Series represents a single column with index, which is either independent or belongs to a Pandas DataFrame.
- We can create a series from **list, tuple, dictionary and numpy array.**
- To create a series first we have to import the pandas library.
- `>>> import pandas as pd`
- `>>> pd.Series([1,2,3,4,5])  #from list`
- `>>> pd.Series(('a','b',20,7.5)) #from tuple`
- `>>> pd.Series({'a':12,'b':23,'c':'AB','d':'OX'}) #from dictionary`
- `>>> pd.Series(np.array([11,20,12,32,52]))  #from numpy array`

PANDAS DATAFRAME

- Pandas DataFrame is a widely used data structure which works with a two-dimensional array with labeled axes (rows and columns).
- We can create DataFrame using list, tuple, dictionary, numpy array and pandas series.
- To create a data frame first
- `import pandas as pd`
- `>>> pd.DataFrame([[1,2,3],[11,22,33],[111,222,333]]) #using list`
- `>>> pd.DataFrame(((10,20,30),(21,22,23),(74,52,25))) #using tuple`
- `>>> pd.DataFrame({'a':[5,2],'b':[4,3],'c':[9,5]}) #using dictionary`
- `>>> pd.DataFrame([np.array([5,2,3]),np.array([1,6,4]),np.array([0,7,2])])`
- `>>> pd.DataFrame([pd.Series([4,7,1]),pd.Series([3,5,1]),pd.Series([8,7,1])] )`

BASIC FUNCTIONALITY AND ATTRIBUTES OF DATAFRAME

DataSpace ACADEMY

- `>>> df=pd.DataFrame({'a':[5,2,9],'b':[4,3,8],'c'[9,5,7],'d'['x','y','z']})`
- `df.head()`
  - Returns first five records.
- `df.tail()`
  - Returns last five records.
- `df.sample()`
  - Returns only one random sample.
- `df.info()`
  - Returns the basic information of the dataset.
- `df.describe()`
  - Returns the description of the data.

BASIC FUNCTIONALITY AND ATTRIBUTES OF DATAFRAME CONTD.

- `df.shape`
  - Returns the DataFrame shape
- `df.dtyes`
  - Returns the columns data type.
- `df.index`
  - Returns a RangeIndex object with the start, stop, and step values.
- `df.values`
  - Returns all the values in the DataFrame.
- `df.columns`
  - Returns all the columns name.

CHECKING & HANDLING DUPLICATES

DataSpace ACADEMY

- `>>> data = {'score':[98,98,48,75,75,96,96], 'stream':['EE','EE','IT','CS','CS','MCA','MCA'] }`
- `>>> df2 = pd.DataFrame(data)`
- `df2.duplicated()`
  - Duplicated values are indicated as True values in the resulting Series.
- `df2.duplicated().sum()`
  - Returns the sum of total duplicates
- `df2.drop_duplicates(subset=['col'],inplace=True)`
  - Return DataFrame with duplicate rows removed.

CHECKING & HANDLING NULL VALUES

DataSpace ACADEMY

- `>>> data2 = {'score':[98,np.nan,48,75,np.nan,96,np.nan], 'stream':['EE',np.nan,'IT',np.nan,'CS','MCA',np.nan], 'pincode':[np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.nan] }`
- `>>> df3 = pd.DataFrame(data2)`
- `df3.isna()`
  - Detect missing values. Show which entries in a DataFrame are NA.
- `df3.isnull()`
  - Detect missing values. Show which entries in a DataFrame are NA.
- `df3.isnull().sum()`
  - Returns the total number of null values in each columns.
- `df3.fillna(value=None, *, method=None, axis=None, inplace=False)`
  - Fill NA/NaN values using the specified method.

DROPPING COLUMNS AND ROWS

- `df3.drop(columns = list_of_cols_to_drop,inplace=True,axis=1)`
  - For permanent drop of columns
- `df3.drop(rows = list_of_rows_to_drop,inplace=True,axis=0)`
  - For permanent drop of rows.
- `df3.dropna(axis= 1 or 0, inplace=True])`
  - Return a new Series with missing values removed.

SELECTING COLUMNS FROM A DATAFRAME

DataSpace ACADEMY

- Create a DataFrame
- `>>> data = {'name':['avi','akshat','ankita','riya','sourav'], 'roll_no':[25,45,20,5,42],'score': [98,48,75,80,96], 'stream':['EE','IT','CS','ECE','MCA'] }`
- `>>> df4 = pd.DataFrame(data)`
- Example:
- First selecting a single column
- `>>> df4['score']`
- Selecting multiple column
- `>>> df4[['name', 'score']]`
- `>>> df4[['stream','roll_no','score']]`

DataSpace ACADEMY

97

MATPLOTLIB & SEABORN

Data Visualization Libraries

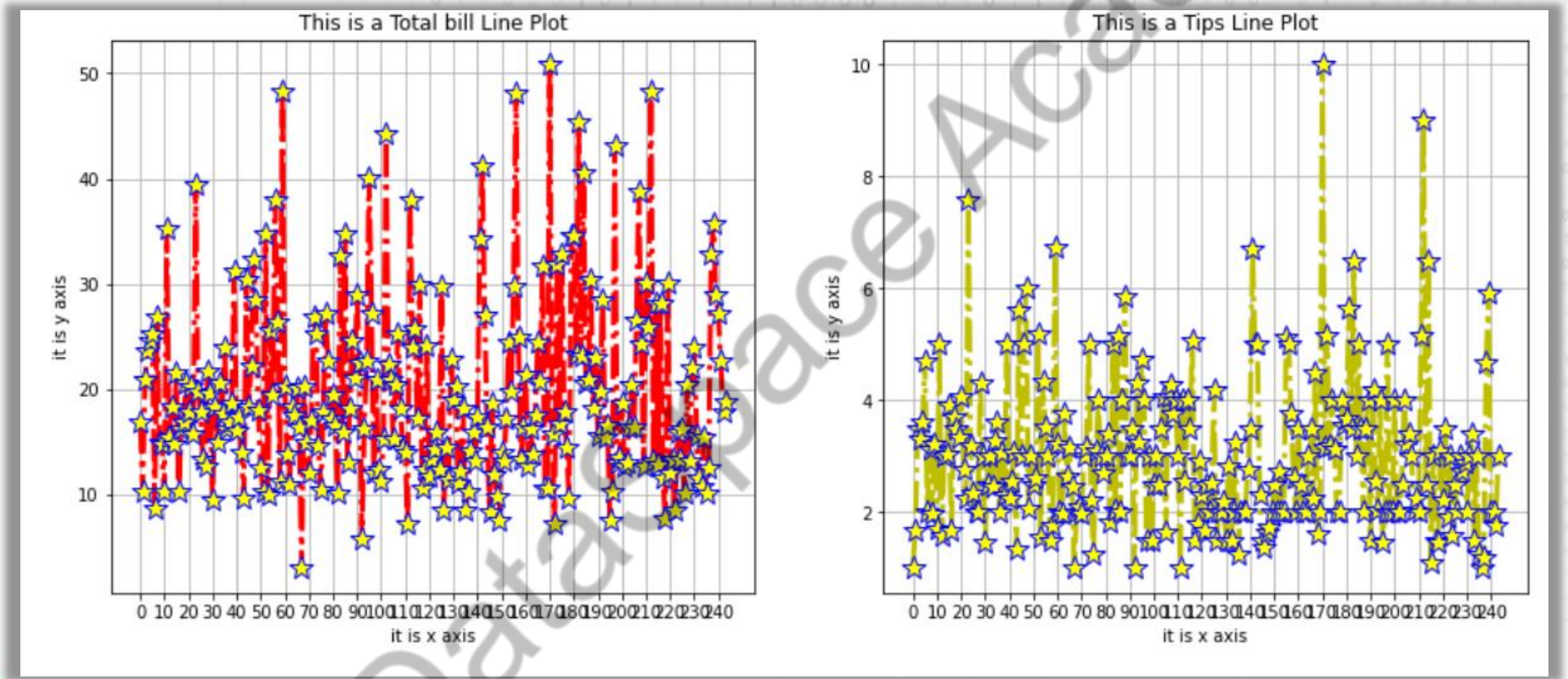
MATPLOTLIB AND SEABORN

- **Matplotlib** is a Python library for creating static, animated, and interactive visualizations.
- **Seaborn** is a Python data visualization library based on Matplotlib, another popular plotting library. Seaborn is specifically designed for creating attractive and informative statistical graphics.

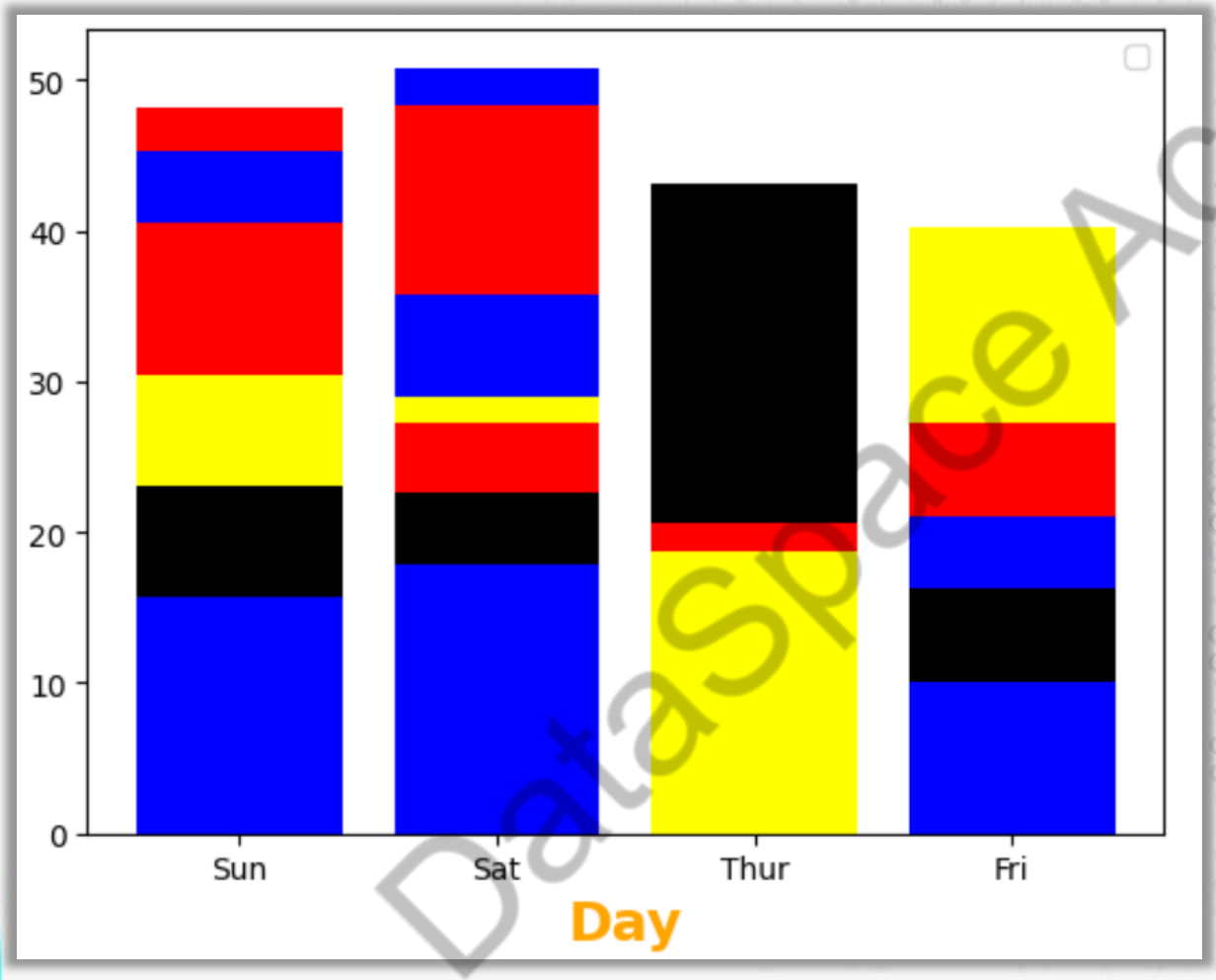
SOME COMMON GRAPH

- Line plot
- Bar graph
- Histogram
- Boxplot
- Scatter plot (need two numerical columns for two axis)
- Stem plot
- violin plot
- Kernel density Estimator(KDE)

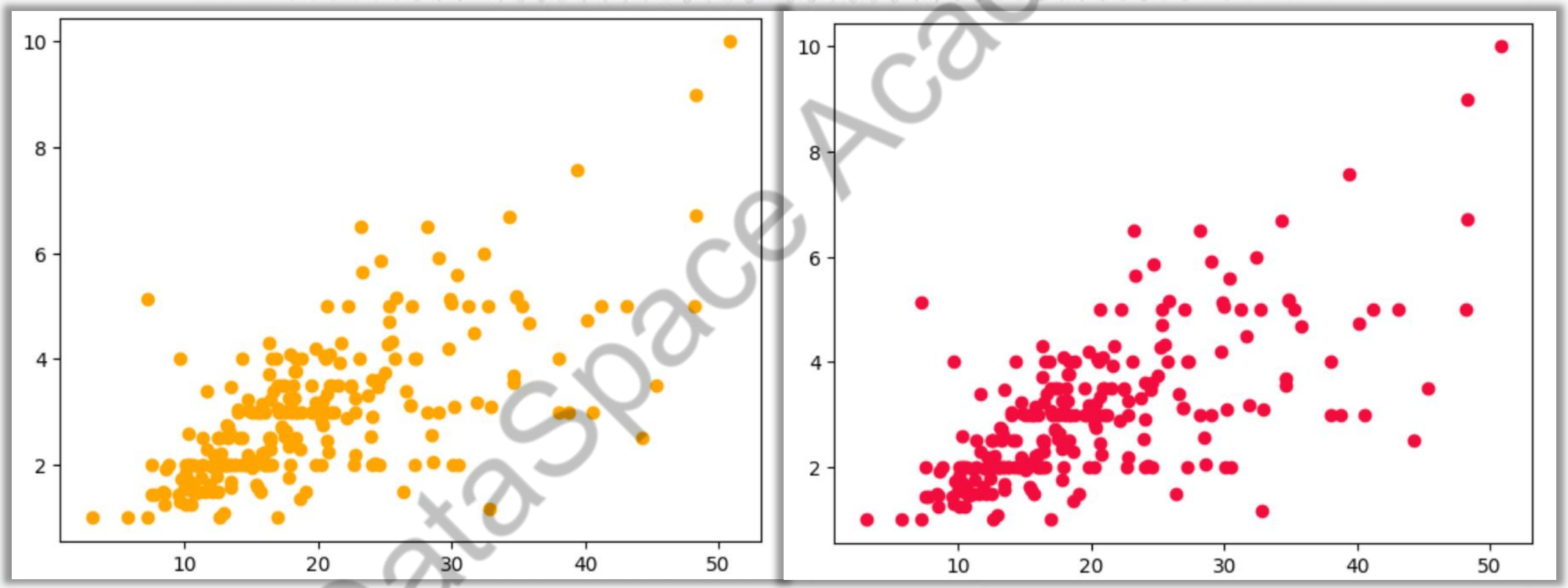
LINE PLOT



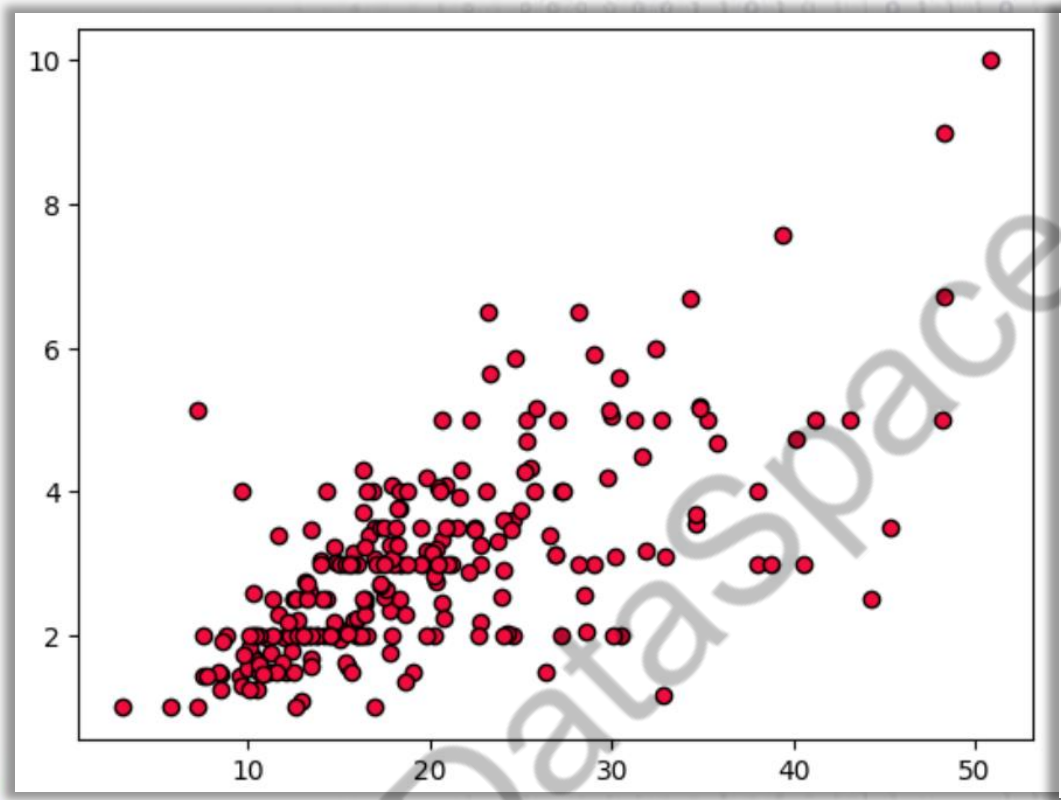
BAR PLOT

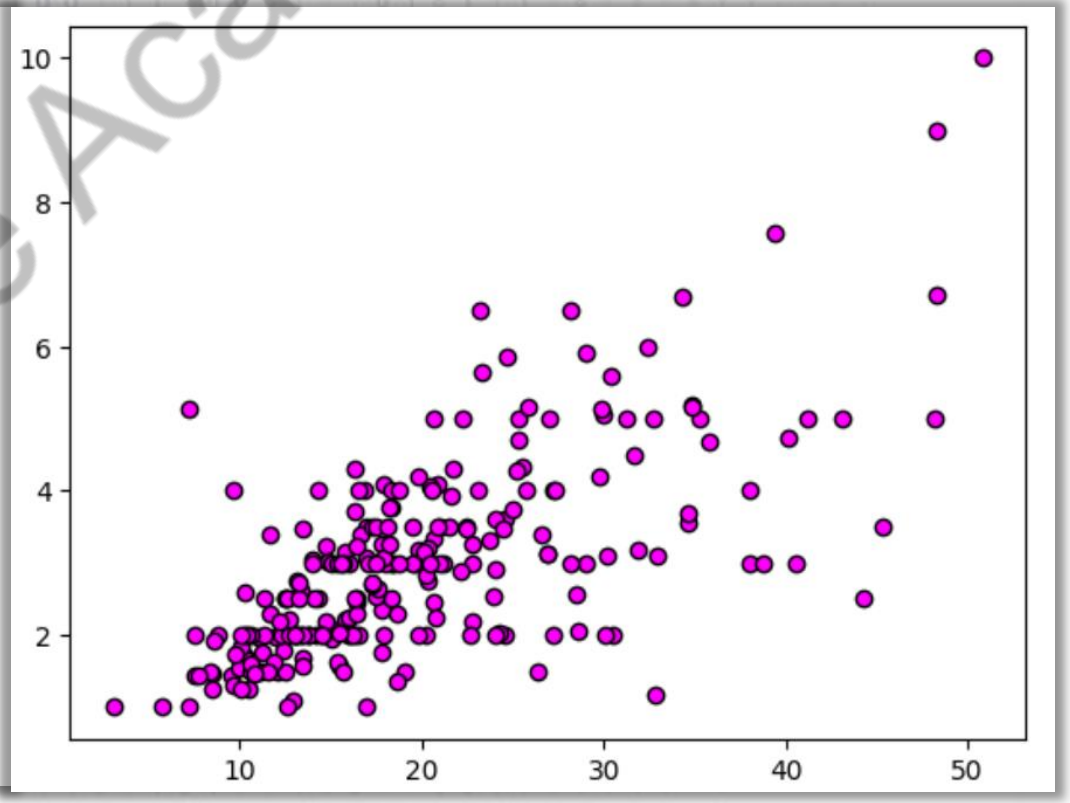


SCATTER PLOT

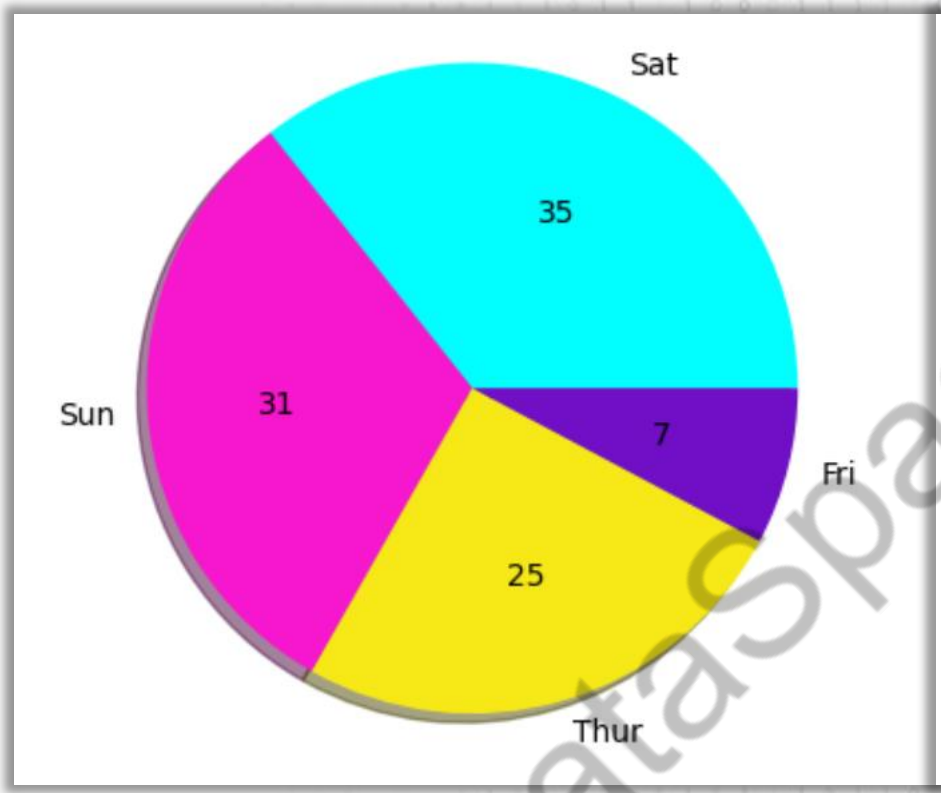


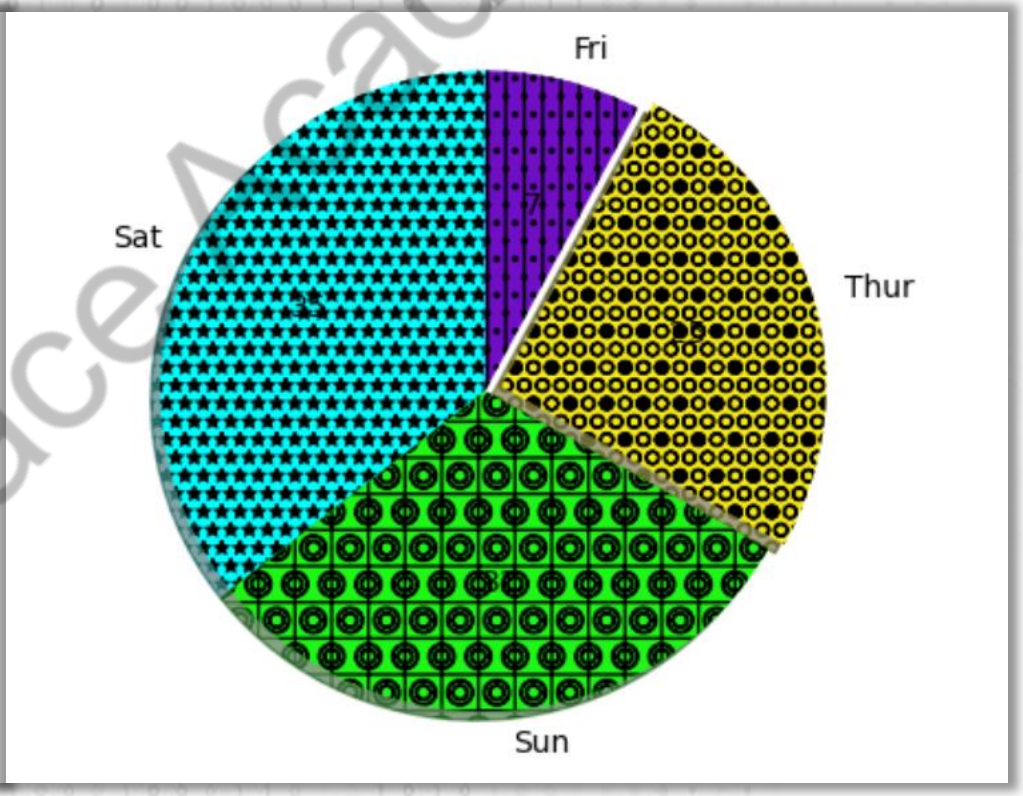
CUSTOMIZE SCATTER PLOT



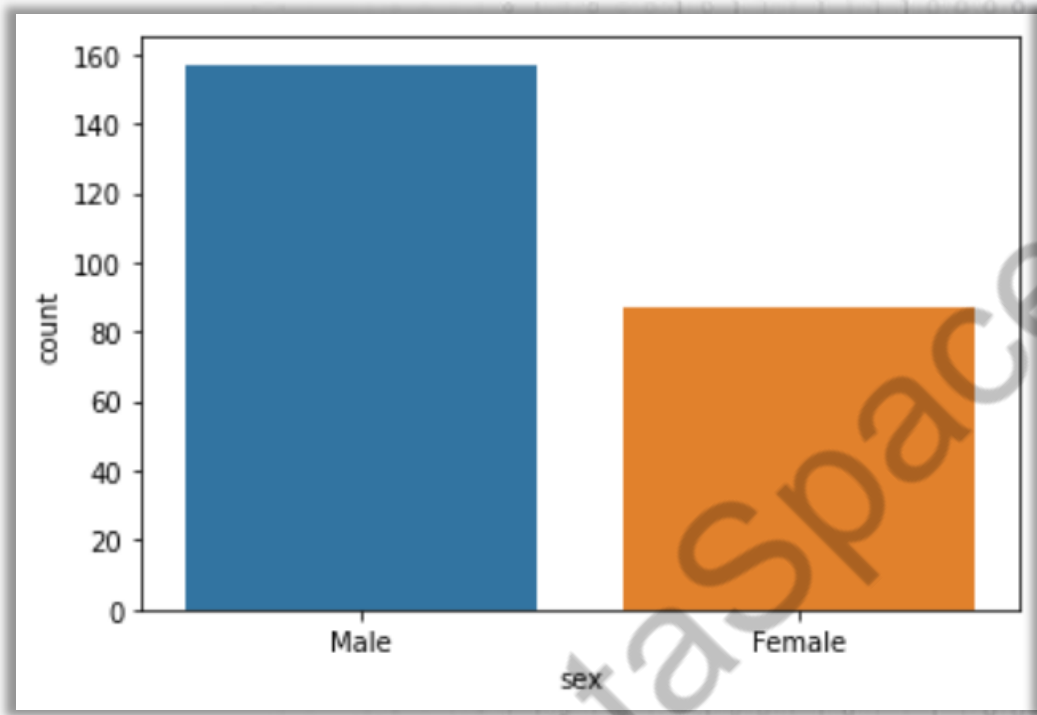


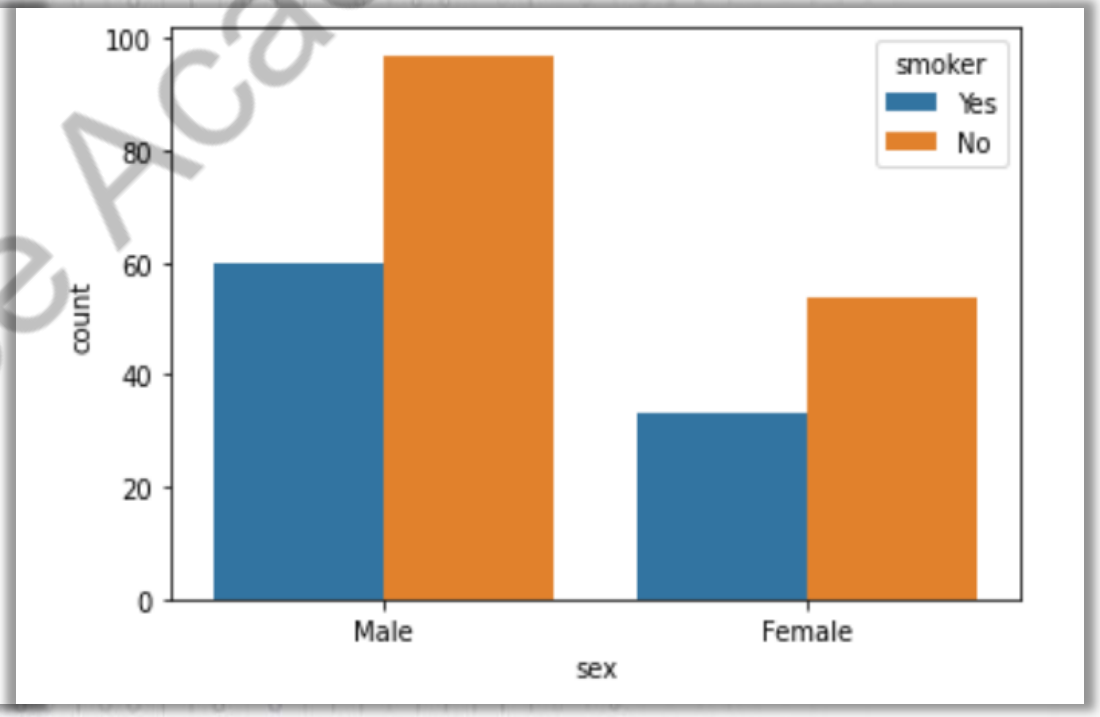
PIE PLOT



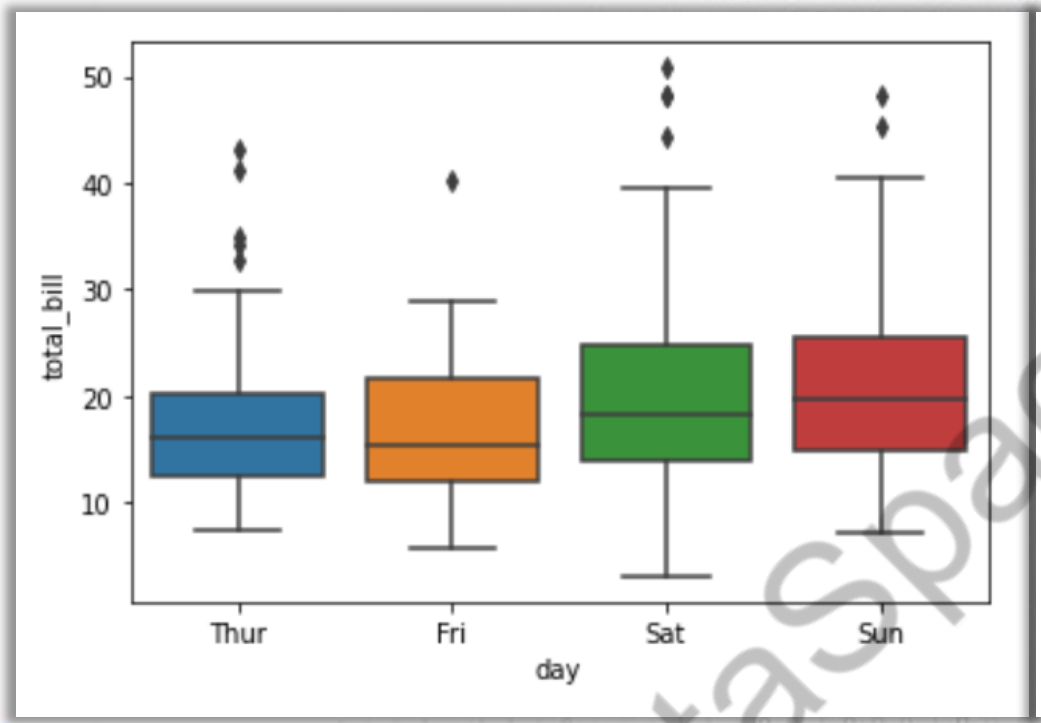


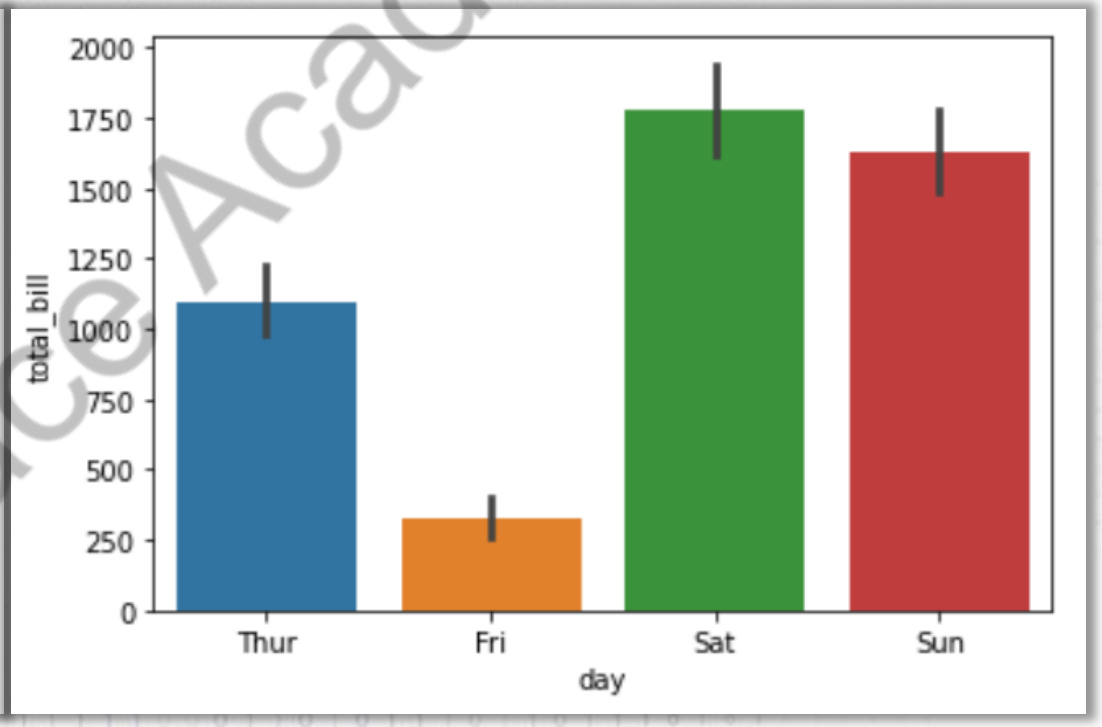
COUNT PLOT



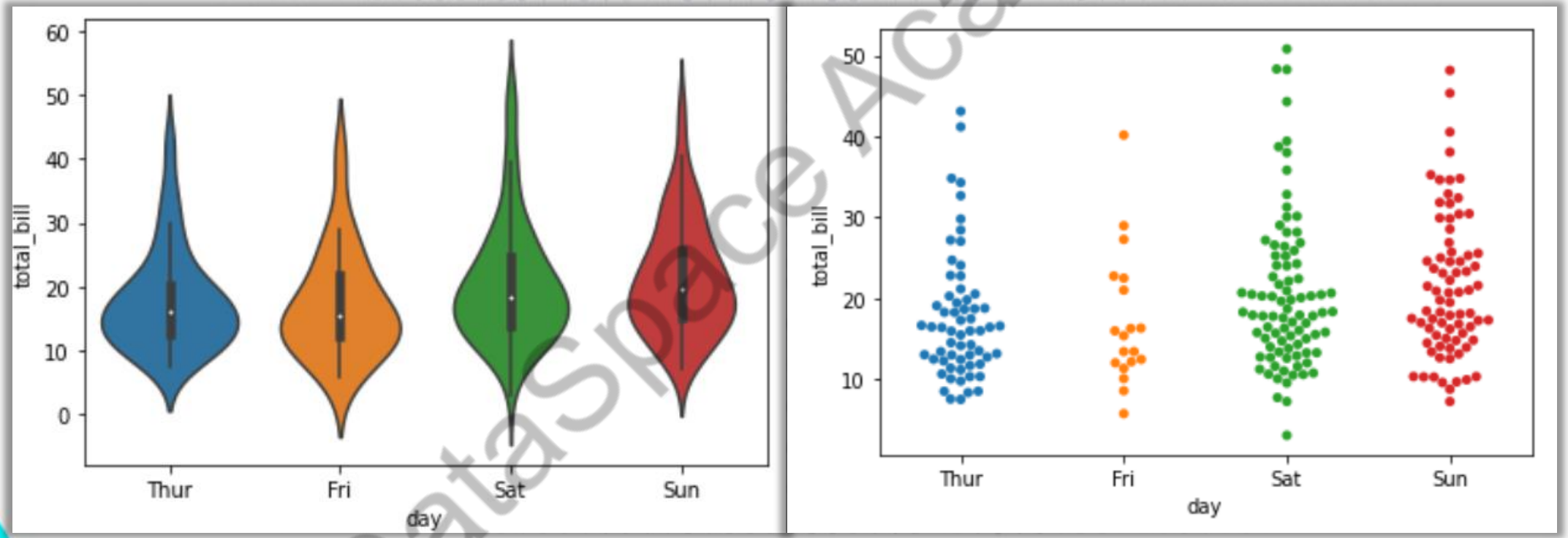


BOX & BAR PLOT

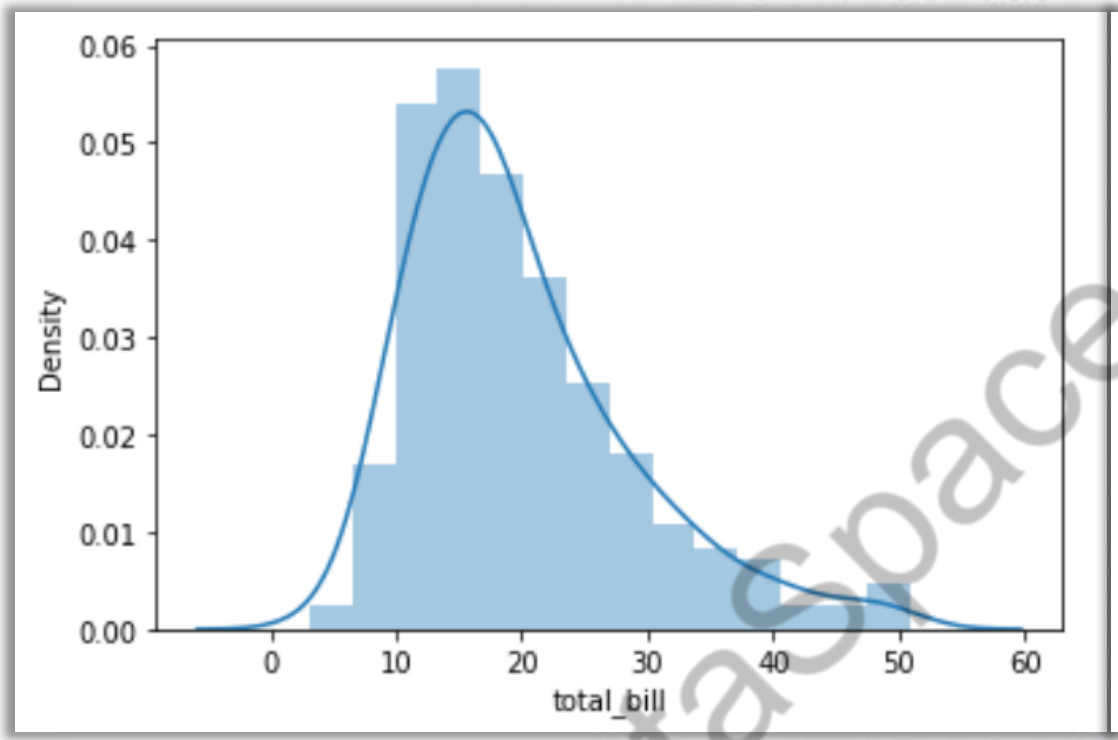


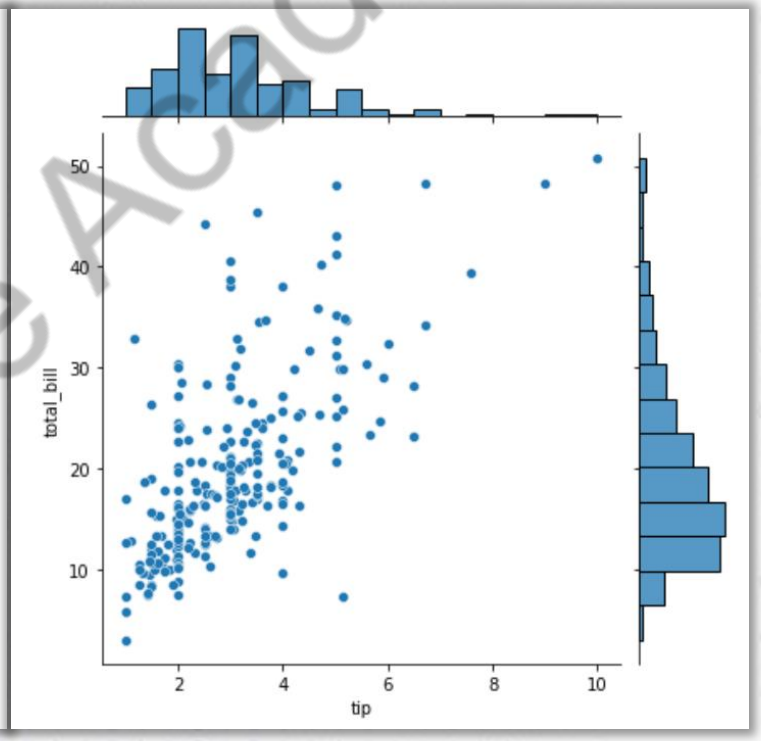


VIOLIN & STRIP PLOT

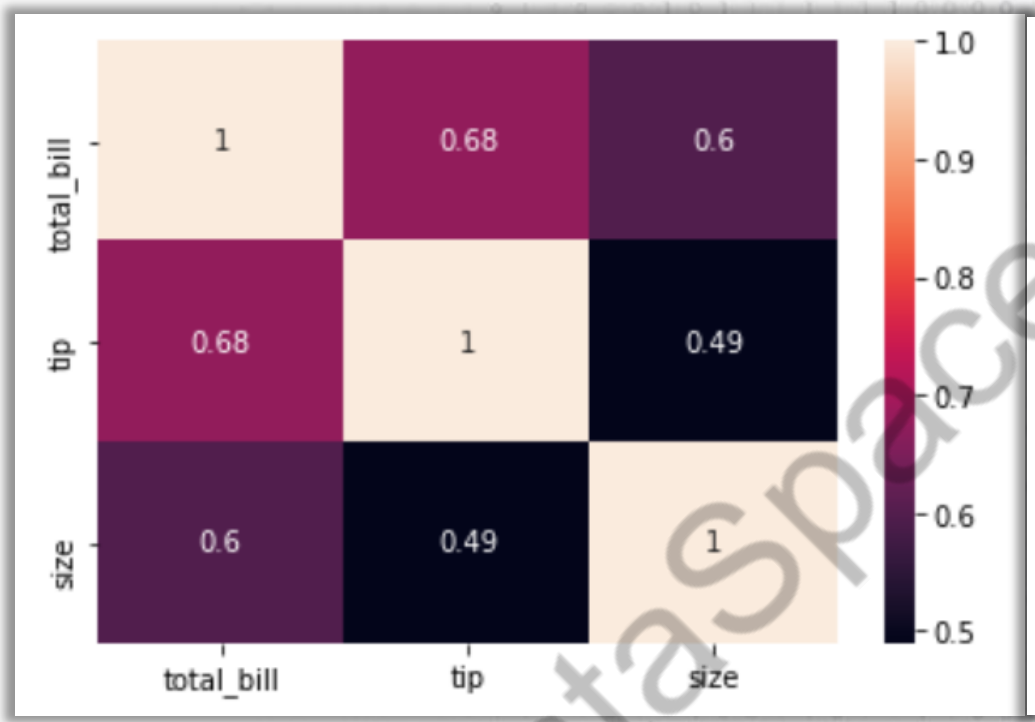


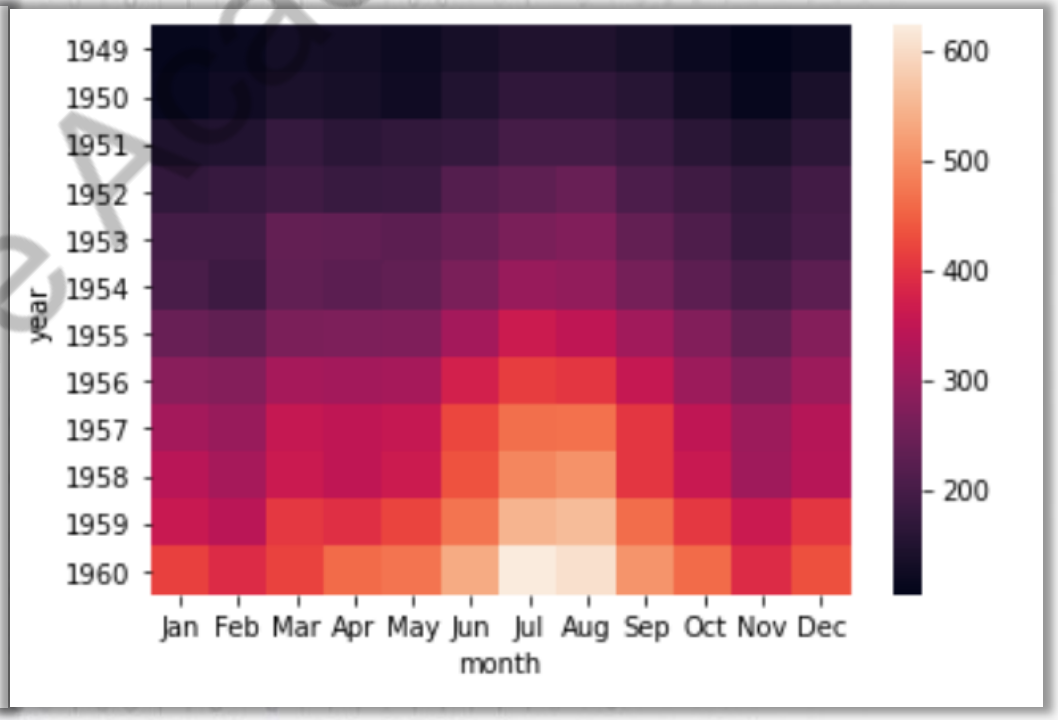
DISTRIBUTION PLOT





HEATMAP





DataSpace ACADEMY

LETS DO CODING

DataSpace ACADEMY

111

MACHINE LEARNING

Welcome to the Machine Learning

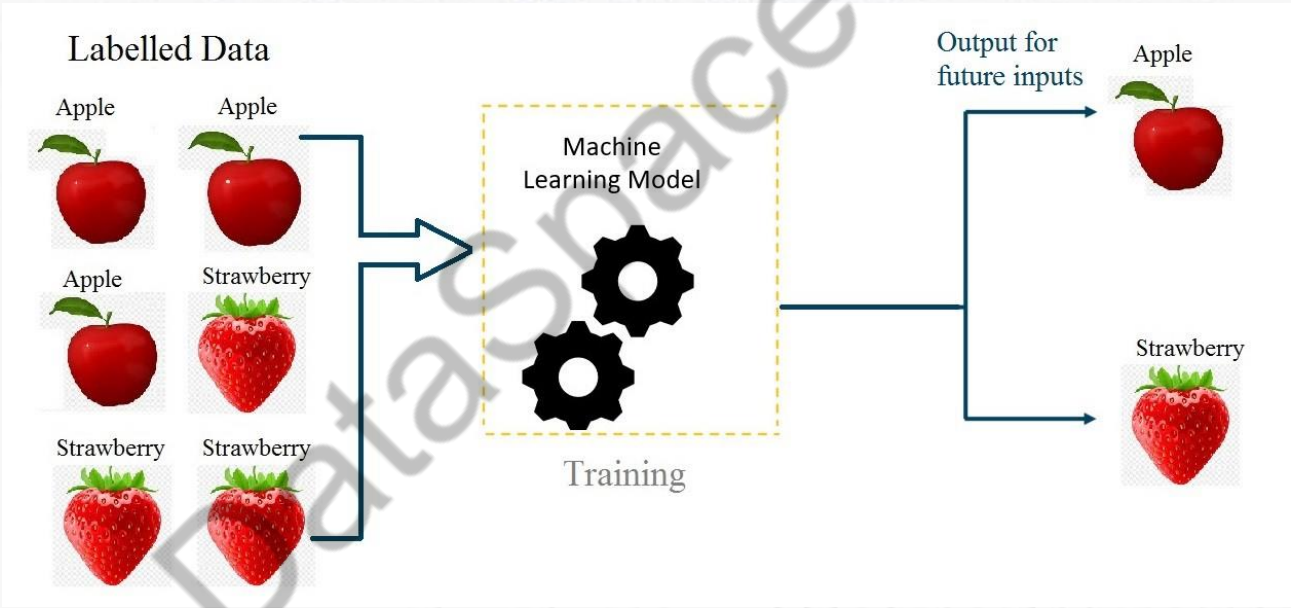
INDEX

TYPES OF MACHINE LEARNING

- Supervised Learning
- Unsupervised Learning
- Semi Supervised Learning
- Reinforcement Learning

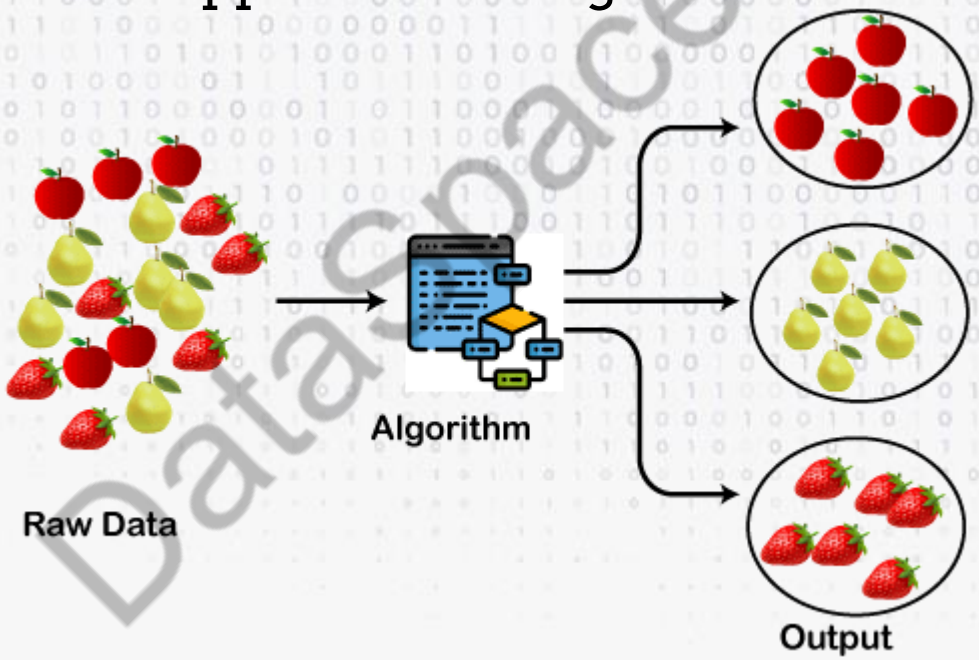
SUPERVISED LEARNING

- Supervised Learning is the one where you can consider the learning is guided by a teacher , we have a dataset which act as a teacher and it's role is to train the model or the machine. Once the model gets trained It can start making a prediction or decision when new data is given to it.

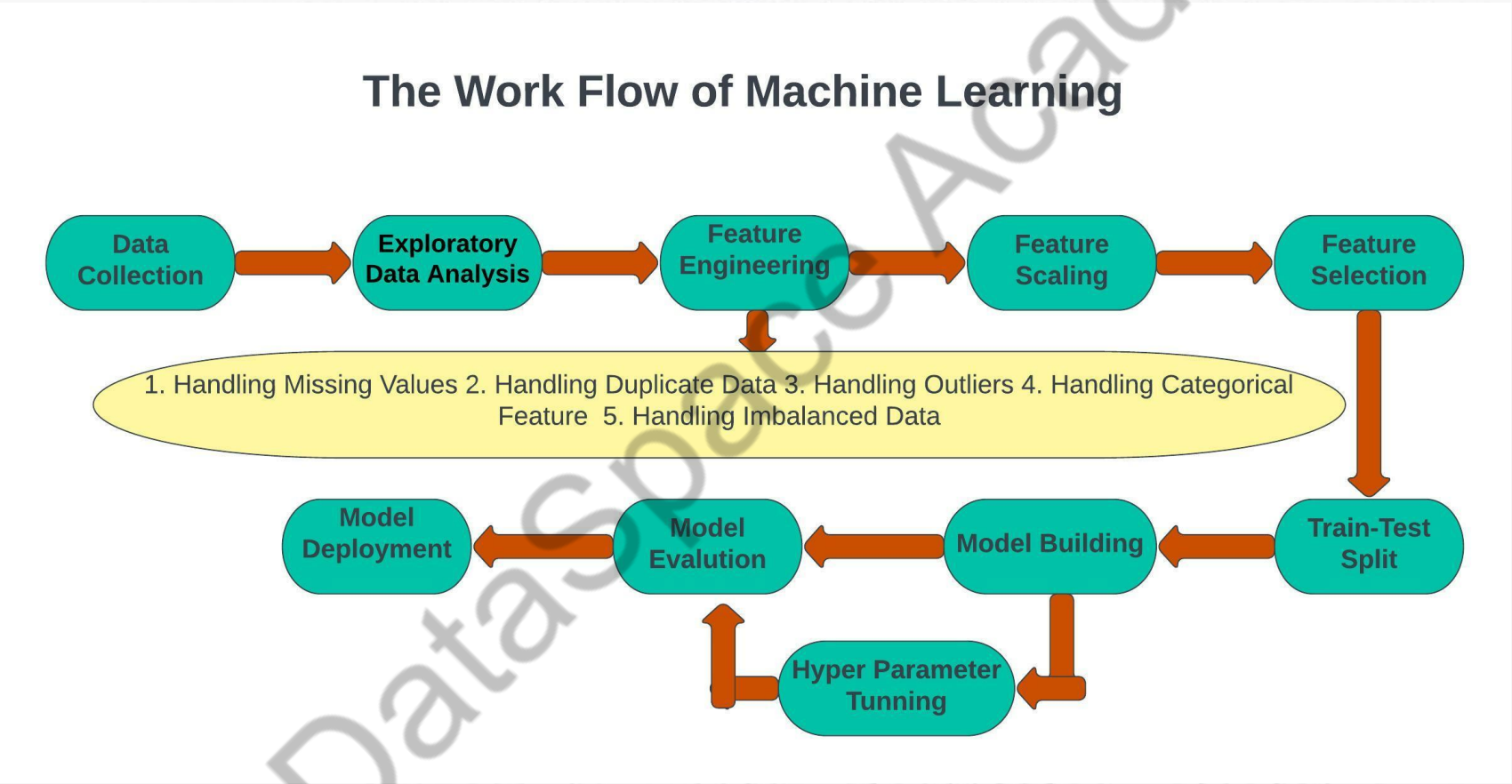


UNSUPERVISED LEARNING

- In unsupervised learning the model learns through observations and finds Structures in the data. Once the model is given a dataset it automatically Finds patterns and relationship in the dataset by creating cluster in it. What it cannot do is and labels to the cluster, like it can not say this a group of apples or mangoes, but it will separate all the apples from mangoes



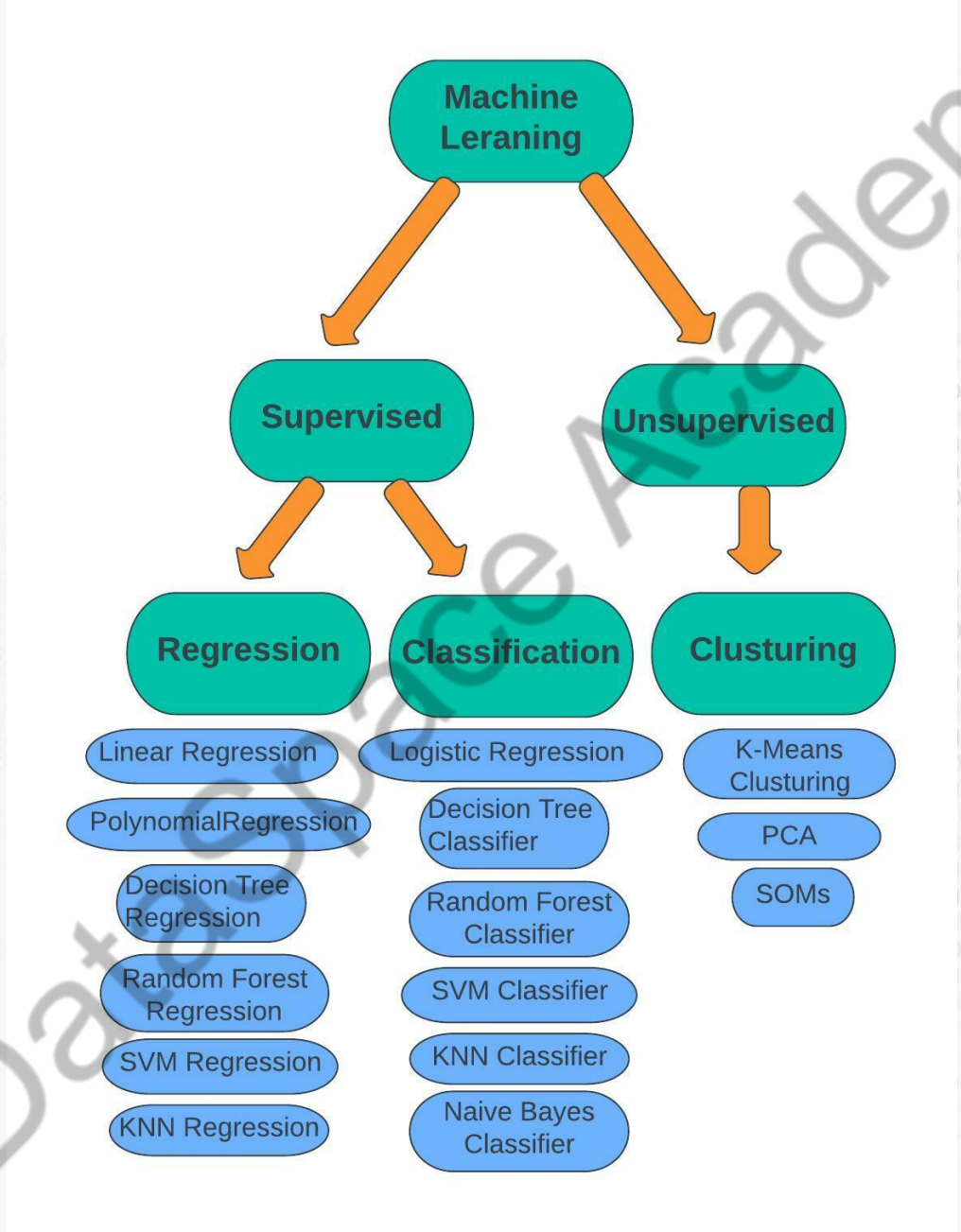
WORK FLOW OF ML



DataSpace ACADEMY

117

ML ALGORITHM



REGRESSION

- **What is Linear Regression ?**
  - Linear regression is a supervised machine learning algorithm that let us understand the relationship between dependent and independent variables. It predicts continuous value.
  - example: y = mx + b

    Here:

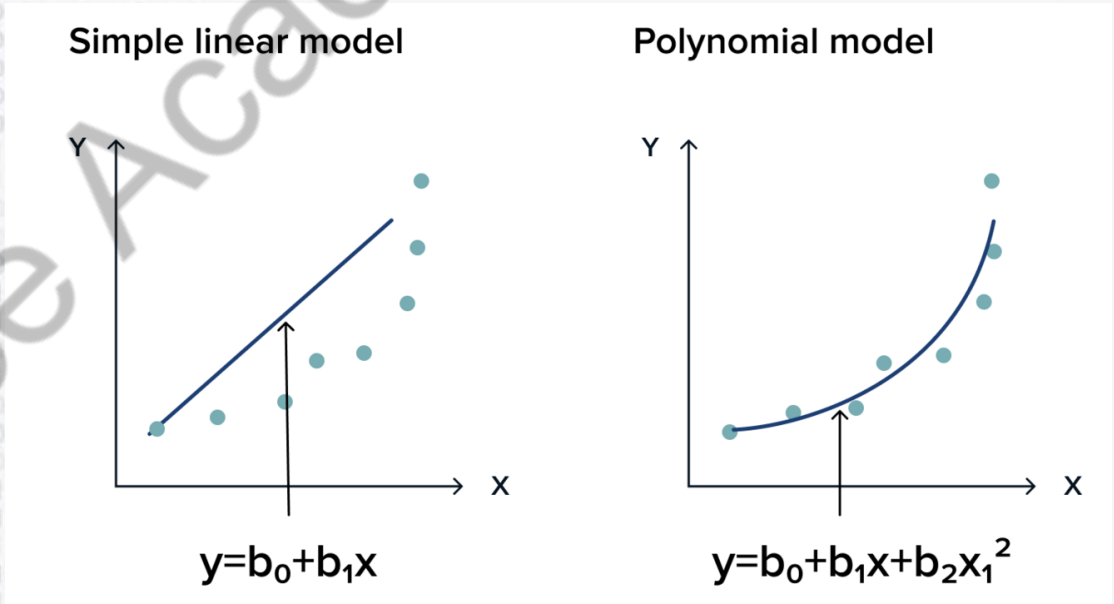
    - x = independent variable
    - y = dependent variable
    - b = bias
    - m = slope

  By training on many datapoints, the model understands value of b and m.

  Then taking any value x, the model can estimate the value y.

TYPES OF REGRESSION

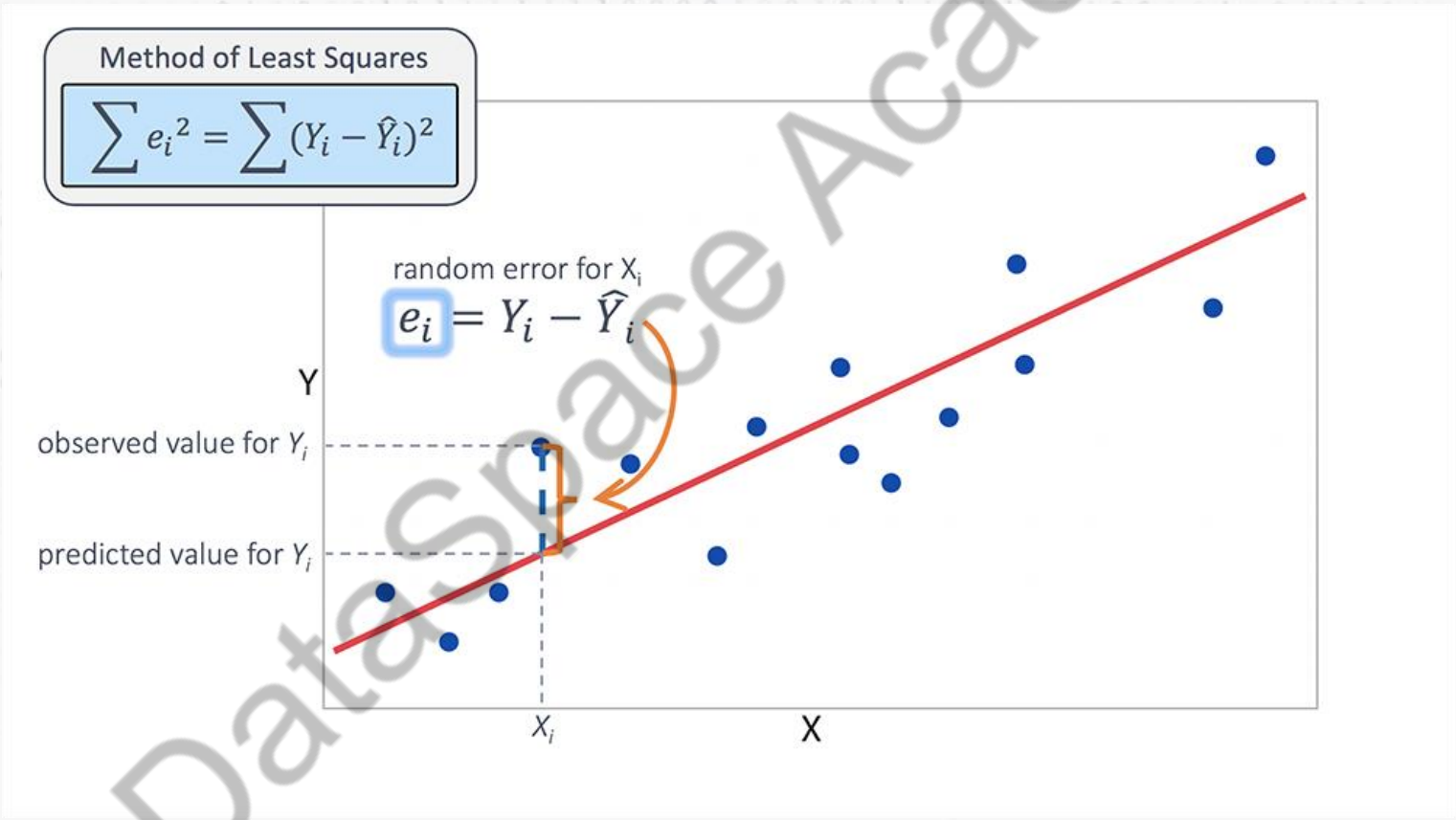
- Linear Regression
- Multiple Linear Regression
- Polynomial Regression



LINEAR VS POLYNOMIAL REGRESSION

Simple Linear Regression	$y = b_0 + b_1 x_1$
Multiple Linear Regression	$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$
Polynomial Linear Regression	$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$

EQUATION OF REGRESSION



LOSS FUNCTION IN REGRESSION

Mean Absolute Error (MAE) is the mean of the absolute value of the errors:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Squared Error (MSE) is the mean of the squared errors:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

LOSS FUNCTION CONTD.

- R-squared and adjusted R-squared are both statistical measures used to evaluate the goodness of fit of a regression model.
- Adjusted R-squared is similar to R-squared, but it penalizes the model for adding additional independent variables.
- Value of R-squared and adjusted R-squared are ranges from 0 to 1.

$$R^2 = 1 - \frac{SS_{residuals}}{SS_{total}}$$

$$\text{Adjusted } R^2 = 1 - \frac{SS_{residuals} / (n-K)}{SS_{total} / (n-1)}$$

Practical Implementation

DataSpace ACADEMY

126

CLASSIFICATION TASK

Machine Learning

DataSpace ACADEMY

127

LOGISTIC REGRESSION

Supervised ML Algorithm

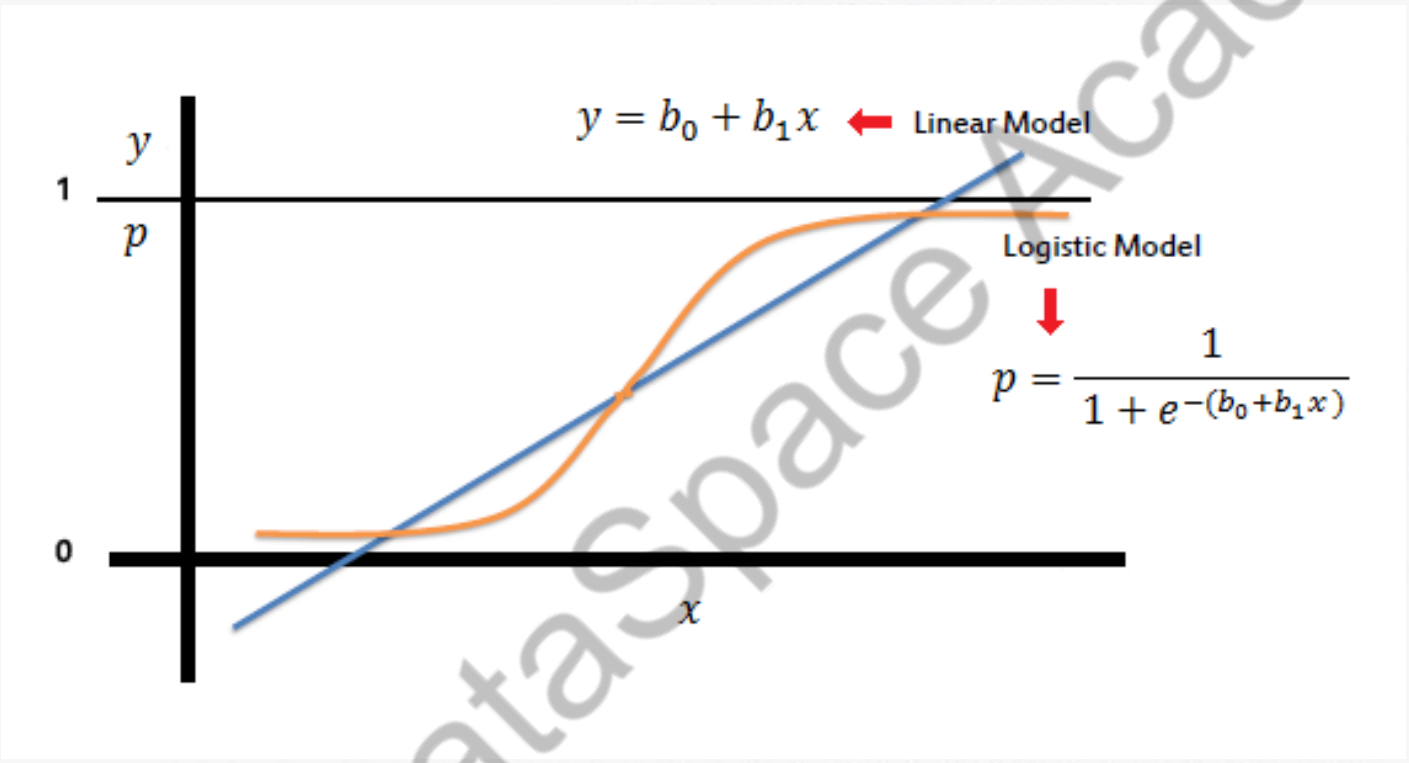
TYPES OF CLASSIFICATION

- Binary Classification, Ex: Yes or No
- Multiclass Classification, Ex: More than two class
- Ordinal Classification, Ex: class in order , Rating(1-5)

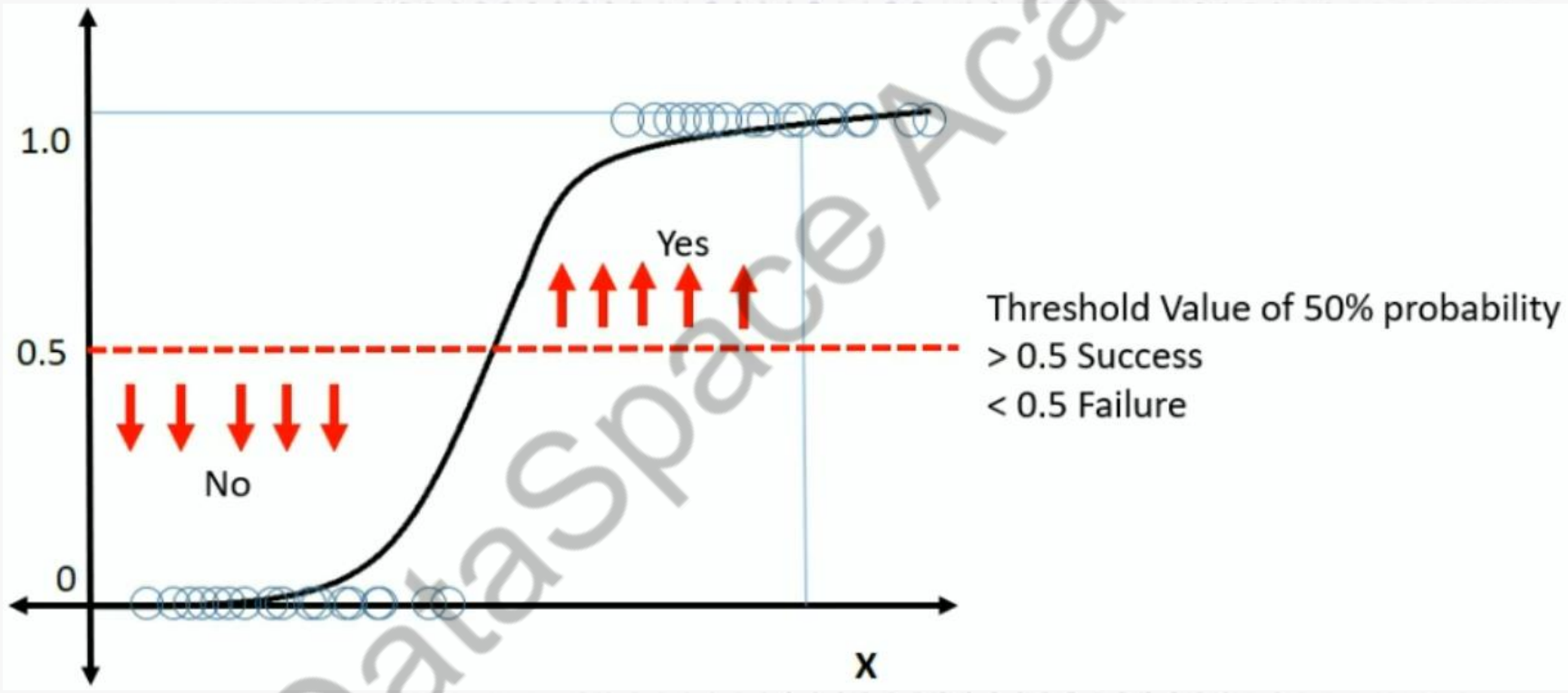
LOGISTIC REGRESSION

- **What is Logistic Regression?**
  - Logistic regression is a supervised machine learning algorithm that is used to predict a binary outcome, such as yes or no, based on prior observations of a data set.
  - Example of Logistic Regression:
    - Predicting whether a customer will default on a loan.
    - Predicting whether or not a patient will have a heart attack.
    - Predicting whether or not an email is a spam.
    - Predicting whether or not a tumor is malignant.

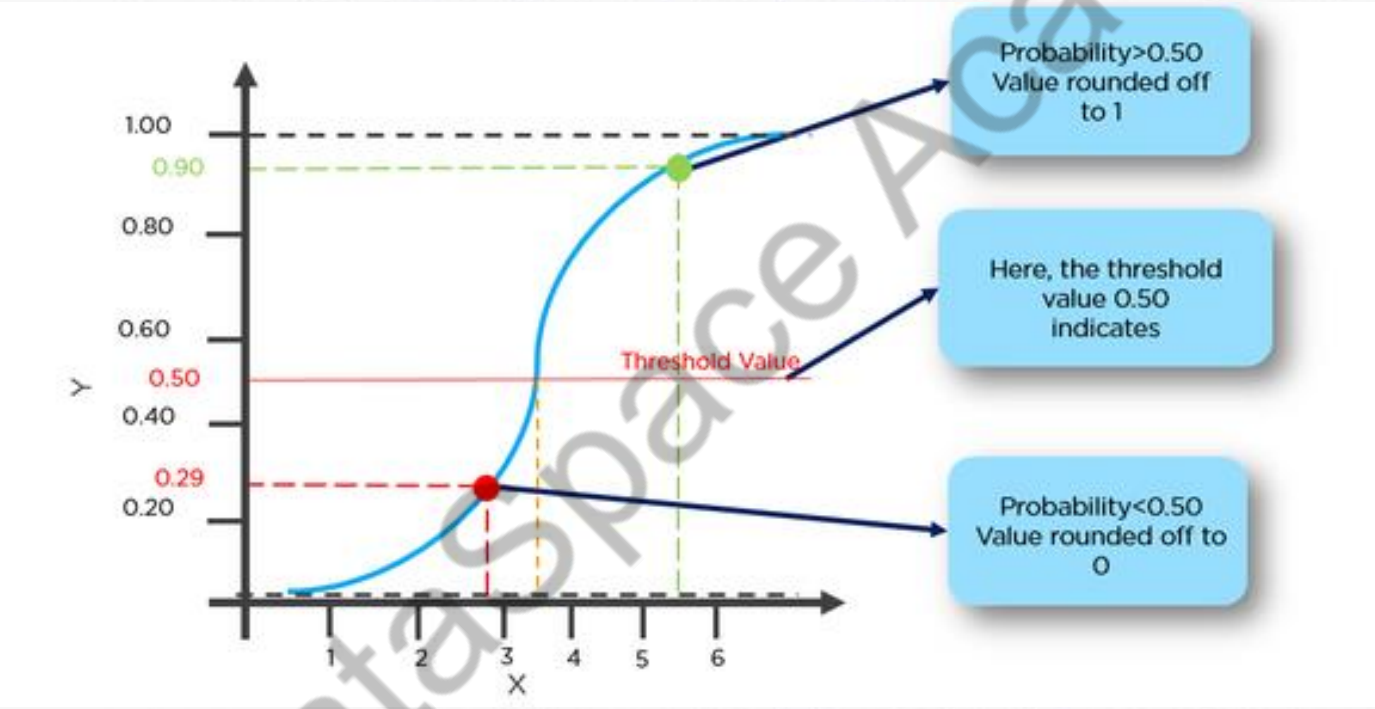
LOGISTIC REGRESSION CONTD.



LOGISTIC REGRESSION CONTD.



LOGISTIC REGRESSION CONTD.



DataSpace ACADEMY

Practical Implementation

DataSpace ACADEMY

DAY 3

DataSpace ACADEMY

135

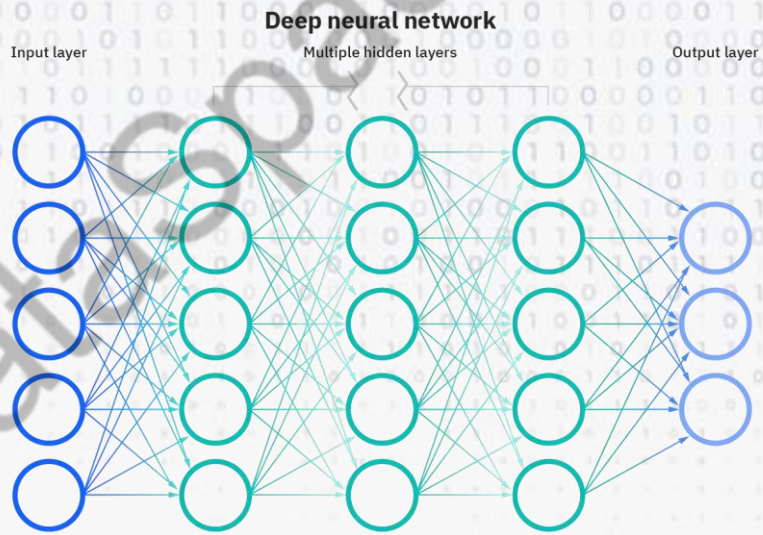
AI & BI

Deep Learning and Power BI

DataSpace ACADEMY

DEEP LEARNING AND NEURAL NETWORK

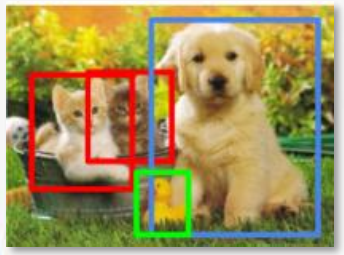
- **Deep learning** is a type of machine learning that uses artificial neural networks to learn from data.
- **Neural networks** are inspired by the human brain, and they are able to learn complex patterns in data that would be difficult or impossible to learn with traditional machine learning methods.

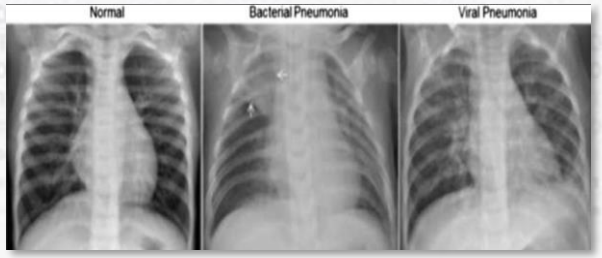


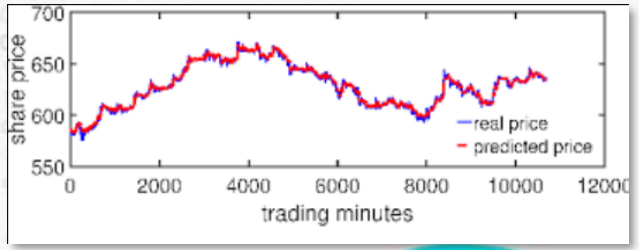
APPLICATIONS OF NEURAL NETWORK

- Here are some examples of deep learning applications:
  - Image classification
  - Natural language processing
  - Speech recognition
  - Medical diagnosis
  - Financial trading

These are just a few examples of the many ways that deep learning is being used today.







DataSpace ACADEMY

Practical Implementation

DataSpace ACADEMY

139

POWER BI

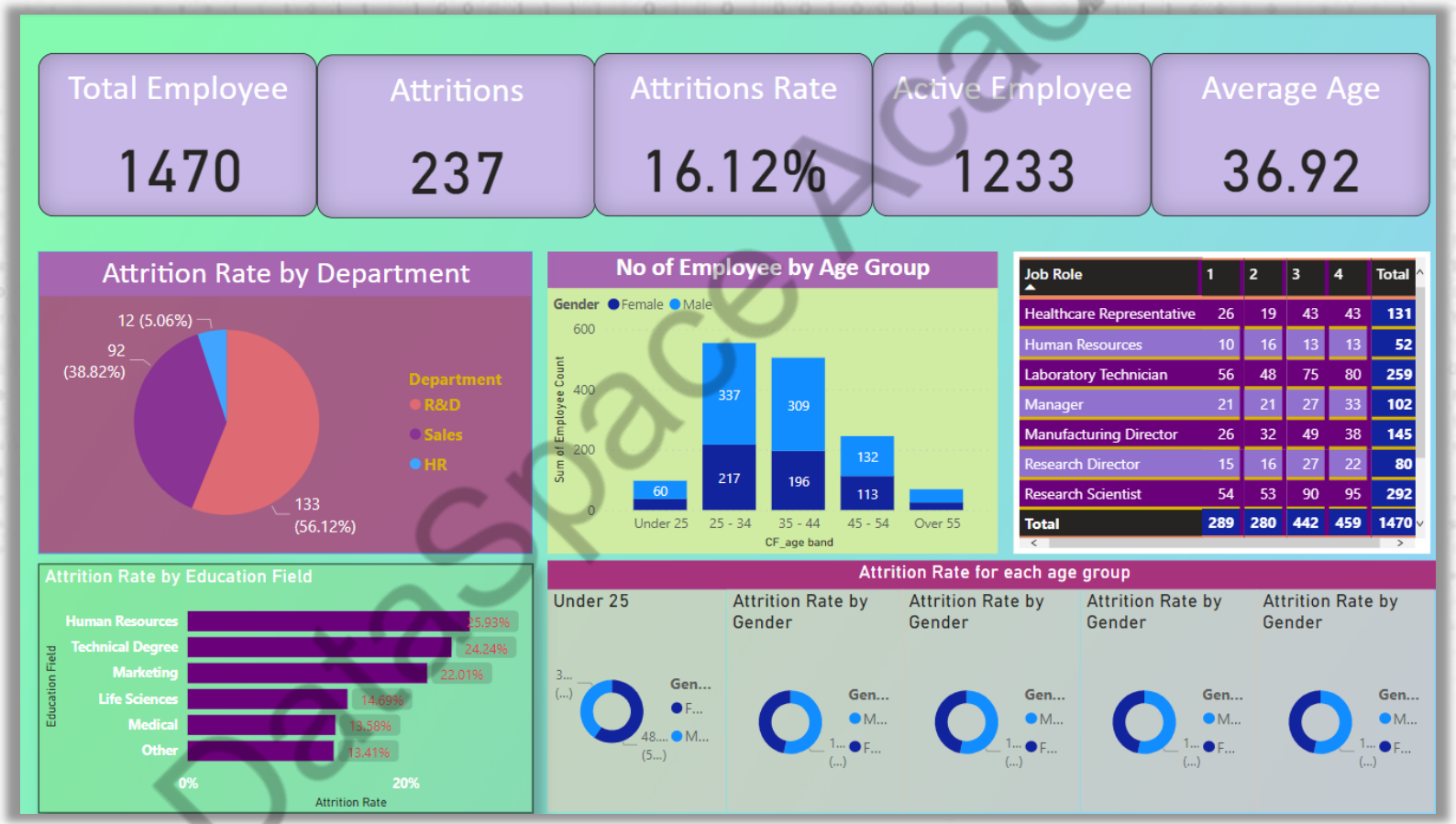
POWER BI

- **Power BI** is a business intelligence (BI) tool that helps you collect, analyze, and visualize data.
- With Power BI, you can connect to a variety of data sources, including Excel spreadsheets, databases, csv and many more.
- We can use Power BI's powerful analysis tools to clean, transform, and model your data.
- **Dashboard** is a visual representation of data that helps you track and analyze key metrics. You can create dashboards by combining reports, tiles, and other elements.

POWER BI CONTD.

- Power BI has a wide variety of charts that you can use to visualize your data.
- Some of the most common chart types include:
  - Bar charts:
  - Column charts:
  - Line charts
  - Area charts
  - Pie charts:
  - Doughnut charts
  - Scatter charts:
  - Bubble charts
  - Animated Bar Chart

DASHBOARD



DASHBOARD

DataSpace ACADEMY

Thank you